NOTE TO USERS

The original manuscript received by UMI contains pages with indistinct print. Pages were microfilmed as received.

This reproduction is the best copy available

UMI

University of Alberta

Analysis of Power Dissipations in CMOS Circuit Designs

By

Nelson Lawrence Rodnunsky

©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Science.

Department of Electrical and Computer Engineering

Edmonton, Alberta

Fall 1998

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisitions et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-34409-6

Canada

University of Alberta

Faculty of Graduate Studies and Research

The Undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Analysis of Power Dissipations in CMOS Circuit Designs** submitted by Nelson Lawrence Rodnunsky in partial fulfillment of the requirements for the degree of Master of Science.

Dr. N. G. Durdle (Supervisor)

Dr. X. Sun

Dr. J. L. Pinfold

Date: 19 May 98

Abstract

Much of the recent work in VLSI is the estimation of power dissipation by a circuit design. This research introduces a new analysis software called ***power***. It was designed to determine and analyze dynamic power dissipations of CMOS circuits prior to fabrication. The short circuit power contribution of a design was also examined. Pattern dependent analysis was used to simulate the effects of the powers dissipated by the circuit. The results are stored in a Matlab compatible format. Source code for a Matlab program is appended to the data file, enabling an interactive, graphical environment for viewing the results.

The results are consistent with that of theoretical expectations and comparisons with other simulation software and analysis techniques available. The program can be used in applications such as fault detection and as an inexpensive analysis tool for university, providing insight into power dissipation considerations when designing circuits for fabrication.

This thesis is dedicated to my parents, ***Isabel and Harold Rodnunsky***, for without whom I would not be here. Their love, dedication, and moral support during the last 29 years will always be remembered.

I also dedicate this to my three best friends: Andrea Hayes, Tania Nichiporik, and Dr. Howard Bennett. Their friendship has been extraordinary and I thank them for being apart of my life.

Acknowledgements

I would like to take this opportunity to thank my supervisor and mentor Dr. Nelson Durdle, whose help and support have been fantastic during the last 3 years. Honorable mention goes to Dr. Martin Margala, who played a major role in my completing this degree. I would also like to thank my examining committee, Dr. Sun (Electrical Engineering) and Dr. Pinfold (Physics), for their acceptance of this thesis.

Many people were involved in my university life during the past six years, particularly the last three while completing my masters program. Here is a ***special thank you*** to the University of Alberta Department of Electrical and Computer Engineering: professors, technical support staff and office staff; especially the following people....

- Dr. Haswell, for your help as my undergraduate advisor and employer during my sessional teaching experiences.
- Dr. Lawson and Dr. Fedosejevs, past and current graduate-student advisors respectively, for your help getting me through this program with very few pains.
- Dr. Routledge, for providing me the opportunity to be a sessional instructor the past two years, and for being a friend.
- Dr. Filanovsky, for your encouragement and assistance.
- Michelle Lock, what can I say? Your help and friendship will always be remembered, in particular your advice to: "Just go with it!". I will not forget you.
- The Office Staff: Shirley, Leena, Carla, Maureen, and Nona, for all of your help and for putting up with my pestering.
- Kees, the guru of all Sys-Admins, thanks for lending an ear, being a friend and for putting up with all the trivial UNIX questions for the last 6 years; and
- Don Presakarchuk, for being the best lab technical assistant and friend.

There are some special people, associates and others that I must also thank.....

- Angela and Craig, for being great teaching assistants in the EE570 lab.
- The Hockey Pool Group: Jonathan, Martin, Derek, Wes, and Roger. Thanks for making procrastination a little more interesting, and leaving my wallet full of dust.
- The Research Office Group: Jonathan, Edmond, Jai, and Kevin. Your support, friendship, and help with the simplest of programming questions was the best.
- My many relatives and friends (outside the university walls) for their support and encouragement, and finally,
- To anyone who I failed to mention, you have not been forgotten only temporarily misplaced. ..... Thanks for the memories .....

Table of Contents

1.0 INTRODUCTION ---- 1
1.1 OBJECTIVES ---- 1
1.2 JUSTIFICATION / MOTIVATION ---- 2
1.3 OVERVIEW ---- 3

2.0 BACKGROUND ---- 4
2.1 POWER DISSIPATION ---- 4
2.1.1 Static Power ---- *5*
2.1.2 Dynamic Power ---- *8*
2.1.3 Short-circuit Power ---- *12*
2.1.4 Total Power Dissipation ---- *14*
2.2 EXISTING POWER ANALYSIS SOFTWARE ---- 15
2.2.1 Description of Existing Software ---- *15*
2.2.2 Thesis Software Program vs. Existing Software. ---- *16*
2.3 CAD SOFTWARE COMPONENTS ---- 19
2.4 CIRCUIT UNDER TEST (CUT) ---- 20

3.0 MODELS AND ANALYSIS TECHNIQUES ---- 23
3.1 MATHEMATICAL ANALYSIS METHODS AND MODELS ---- 23
3.1.1 Pattern Dependency Vs Probabilistic Analysis Methods ---- *23*
3.1.2 Mathematical Models ---- *24*
3.1.2.1 Capacitance Models ---- 24
3.1.2.2 Node Voltage Models ---- 26
3.1.2.3 Power Dissipation Models ---- 28
3.2 CIRCUIT ANALYSIS TECHNIQUES AND DISCUSSION ---- 31

4.0 SUBROUTINE DESCRIPTION AND SOFTWARE APPLICATION ---- 35
4.1 "POWER" MAIN PROGRAM ROUTINE ---- 35
4.2 LEVEL-ONE SUBROUTINES AND FUNCTIONS ---- 37
4.2.1 "read_models" Subroutine ---- *37*
4.2.2 "read_netlist" Subroutine ---- *38*
4.2.3 "cap_calcs" Subroutine ---- *40*
4.2.4 "combine_subccts" Subroutine ---- *40*
4.2.5 "free_mem" Subroutine ---- *41*
4.2.6 "time_node_voltage" Subroutine ---- *42*
4.3 LEVEL-TWO SUBROUTINES AND FUNCTIONS ---- 44
4.3.1 "file_check" Function ---- *44*
4.3.2 "read_model_data" Subroutine ---- *44*
4.3.3 "lower_case" Subroutine ---- *45*

4.3.4 "inter_cap_sort" Subroutine ---- 46
4.3.5 "para_cap_calc" Subroutine ---- 46
4.3.6 "combine_cap" Subroutine ---- 47
4.3.7 "combine_mos" Subroutine ---- 48
4.3.8 "combine_que" Subroutine ---- 48
4.3.9 "combine_res" Subroutine ---- 48
4.3.10 "cap_sort" Subroutine ---- 49
4.3.11 "res_sort" Subroutine ---- 49
4.3.12 "rename_refs" Subroutine ---- 50
4.3.13 "node_cap" Subroutine ---- 50
4.3.14 "analyze" Subroutine ---- 51
4.3.15 "power_diss" Subroutine ---- 52
4.4 LEVEL-THREE SUBROUTINES AND FUNCTIONS ---- 53
4.4.1 "strtonum" Function ---- 53
4.4.2 "convert" Subroutine ---- 53
4.4.3 "vj_entry" Function ---- 54
4.4.4 "calculations" Subroutine ---- 54
4.4.5 "data_file_transfer" Subroutine ---- 55
4.5 "DRAIN_SOURCE" LEVEL-FOUR FUNCTION ---- 55
4.6 MATLAB PROGRAM DATA FILE ---- 56

5.0 RESULTS ---- 61
5.1 TEST CIRCUITS ---- 61
5.2 SAMPLE APPLICATION SESSION ---- 61
5.3 DATA RESULTS ---- 69
5.3.1 Total Power Results ---- 69
5.3.2 Dynamic Power Results ---- 72
5.4 PROGRAM EFFICIENCY ---- 74

6.0 CONCLUSIONS, FINDINGS AND RECOMMENDATIONS ---- 76
6.1 CONCLUSIONS ---- 76
6.2 LIMITATIONS ---- 77
6.3 FUTURE REVISIONS ---- 78

BIBLIOGRAPHY ---- 80

APPENDIX A.1 PMOS MODEL LIBRARY FILE ---- 82

APPENDIX A.2 NMOS LIBRARY FILE ---- 84

APPENDIX A.3 NPN LIBRARY FILE ---- 86

APPENDIX B.1 TYPEDEFS.H HEADER FILE INFORMATION ---- 87

APPENDIX B.2 COMBINE.H HEADER FILE INFORMATION ---- 92

APPENDIX C.1 PROGRAM BLOCK FLOW DIAGRAM----------94
APPENDIX C.2 LEVEL – ONE PROGRAM FLOWCHARTS ----------96
APPENDIX C.3 LEVEL – TWO PROGRAM FLOWCHARTS---------- 100
APPENDIX C.4 LEVEL – THREE AND FOUR PROGRAM FLOWCHARTS ---------- 112
APPENDIX C.5 MATLAB PROGRAM SOURCE CODE---------- 115
APPENDIX D.1 T-CELL XNOR NETLIST FILE---------- 120
APPENDIX D.2 T-CELL XNOR SCHEMATIC DIAGRAM---------- 122
APPENDIX D.3 K-CELL COMPRESSOR NETLIST FILE ---------- 123
APPENDIX E.1 UNIX MANUAL (MAN) PAGE---------- 126

List of Tables

TABLE 4.1: SAMPLE INPUT VOLTAGES FROM NETLIST FILE ----------43
TABLE 4.2: SAMPLE VOLTAGE ARRAYS FROM THE TIME_NODE_VOLTAGE SUBROUTINE ----------43

TABLE B.1: NETLIST LINKED LIST DATA STRUCTURES.----------87
TABLE B.2: MODEL LINKED LIST DATA STRUCTURES.----------89
TABLE B.3: FINAL CIRCUIT LINKED LIST DATA STRUCTURES.----------92

List of Figures

FIGURE 2.1: EXAMPLE OF A SIMPLE CMOS INVERTER. ---- 5
FIGURE 2.2: EXAMPLE OF A CMOS INVERTER SHOWING LEAKAGE CURRENTS. ---- 6
FIGURE 2.3: (A) TRANSFER FUNCTION AND (B) NOISE MARGIN CHART OF A CMOS INVERTER. ---- 9
FIGURE 2.4: CMOS INVERTER (A) WITH A LOGIC 1 TO 0 STEP, AND (B) WITH A LOGIC 0 TO 1 STEP. ---- 10
FIGURE 2.5: MODEL FOR SHORT-CIRCUIT CURRENT AND INPUT SWITCHING WAVEFORM. ---- 13
FIGURE 2.6: PROCEDURAL BLOCK DIAGRAM OF POWERMILL SOFTWARE PROGRAM. ---- 18
FIGURE 2.7: PROCEDURAL BLOCK DIAGRAM OF THESIS SOFTWARE PROGRAM. ---- 18
FIGURE 2.8: (A) A SIMPLE CMOS NAND GATE, AND (B) A DOUBLE-PASS CMOS NAND GATE. ---- 21
FIGURE 2.9: A 4→2 COMPRESSOR CIRCUIT. ---- 22
FIGURE 3.1: (A) STANDARD CMOS CONFIGURATION, AND (B) NON-STANDARD CMOS CONFIGURATION. ---- 27
FIGURE 3.2: BLOCK DIAGRAM OF OVERALL ANALYSIS LOOP. ---- 32
FIGURE 3.3: BLOCK DIAGRAM OF INTERNAL NODE ANALYSIS FROM "TIME_NODE_VOLTAGE" SUBROUTINE. ---- 34
FIGURE 4.1: BLOCK DIAGRAM OF MAIN PROGRAM "POWER" ROUTINE. ---- 36
FIGURE 4.2: LEVEL-ONE PROGRAM SUBROUTINES. ---- 37
FIGURE 4.3: BLOCK DIAGRAM OF "READ_MODELS" SUBROUTINE. ---- 37
FIGURE 4.4: BLOCK DIAGRAM OF "READ_NETLIST" SUBROUTINE. ---- 39
FIGURE 4.5: BLOCK DIAGRAM OF "CAP_CALCS" SUBROUTINE. ---- 40
FIGURE 4.6: BLOCK DIAGRAM OF "COMBINE_SUBCCTS" SUBROUTINE. ---- 41
FIGURE 4.7: BLOCK DIAGRAM OF "FREE_MEM" SUBROUTINE. ---- 41
FIGURE 4.8: BLOCK DIAGRAM OF "TIME_NODE_VOLTAGE" SUBROUTINE. ---- 42
FIGURE 4.9: BLOCK DIAGRAM OF "READ_MODEL_DATA" SUBROUTINE. ---- 45
FIGURE 4.10: BLOCK DIAGRAM OF "LOWER_CASE" SUBROUTINE. ---- 45
FIGURE 4.11: BLOCK DIAGRAM OF "PARA_CAP_CALC" SUBROUTINE. ---- 46
FIGURE 4.12: BLOCK DIAGRAMS OF (A) "COMBINE_CAP" (B) "COMBINE_MOS" (C) "COMBINE_QUE" (D) "COMBINE_RES" SUBROUTINES. ---- 49
FIGURE 4.13: BLOCK DIAGRAM OF "POWER_DISS" SUBROUTINE. ---- 52
FIGURE 4.14: BLOCK DIAGRAM OF "CALCULATIONS" SUBROUTINE. ---- 54
FIGURE 4.15: BLOCK DIAGRAM OF THE MATLAB SOURCE CODE OPERATION. ---- 56
FIGURE 4.16: FONT SIZE MENU GUI. ---- 57
FIGURE 4.17: MENU OF NODES TO PLOT GUI. ---- 57
FIGURE 4.18: (A) START NODE MENU GUI, (B) END NODE MENU GUI. ---- 58
FIGURE 4.19: EXIT MENU GUI. ---- 58
FIGURE 4.20: PLOT TYPE MENU GUI. ---- 59
FIGURE 5.1: VOLTAGE TRANSITIONS FOR EACH NODE AND TIME TRANSITION. ---- 62
FIGURE 5.2: PARASITIC ENERGY CONSUMED AT EACH NODE FOR EACH TIME TRANSITION. ---- 63
FIGURE 5.3: INTERCONNECT ENERGY CONSUMED AT EACH NODE FOR EACH TIME TRANSITION. ---- 64
FIGURE 5.4: AVERAGE ENERGY CONSUMED AT EACH NODE DUE TO THE PARASITIC CAPACITANCES. ---- 65
FIGURE 5.5: AVERAGE ENERGY CONSUMED AT EACH NODE DUE TO THE INTERCONNECT CAPACITANCES. ---- 65
FIGURE 5.6: AVERAGE DYNAMIC POWER (PARASITIC CONTRIBUTION) CONSUMED AT EACH NODE. ---- 66
FIGURE 5.7: AVERAGE DYNAMIC POWER (INTERCONNECT CONTRIBUTION) CONSUMED AT EACH NODE. ---- 67
FIGURE 5.8: TOTAL AVERAGE DYNAMIC POWER CONSUMED AT EACH NODE. ---- 68
FIGURE 5.9: NORMALIZED POWER DISSIPATION CURVES FOR STANDARD T-CELL AND K-CELL GATES WITH: (A) 5V SUPPLY VOLTAGE AND (B) 3.3V SUPPLY VOLTAGE. ---- 70

FIGURE 5.10: NORMALIZED POWER DISSIPATION CURVES USING 3.3V AND 5V SUPPLY VOLTAGES FOR: (A) K-CELL ADDER CIRCUIT, (B) K-CELL COMPRESSOR CIRCUIT, (C) T-CELL ADDER CIRCUIT, AND (D) T-CELL COMPRESSOR CIRCUIT. ---- 71
FIGURE 5.11: CONTRIBUTION OF DYNAMIC POWER TO TOTAL POWER DISSIPATION BASED ON THE POWER PROGRAM FOR THE TEST GATES AND CIRCUITS. ---- 73
FIGURE 5.12: TIME/EFFICIENCY COMPARISON FOR AN XNOR CHAIN CIRCUIT OF VARIOUS LENGTHS ---- 75

FIGURE B.1: CCT (SUB-CIRCUIT DATA) LINKED LIST STRUCTURE. ---- 87
FIGURE B.2: CAP (CAPACITOR DATA) LINKED LIST STRUCTURE. ---- 88
FIGURE B.3: MOS (MOSFET DATA) LINKED LIST STRUCTURE. ---- 88
FIGURE B.4: QUE (TRANSISTOR DATA) LINKED LIST STRUCTURE. ---- 88
FIGURE B.5: RES (RESISTOR DATA) LINKED LIST STRUCTURE. ---- 88
FIGURE B.6: SUB (SUB-CIRCUIT REFERENCE DATA) LINKED LIST STRUCTURE. ---- 89
FIGURE B.7: VOL (VOLTAGE DATA) LINKED LIST STRUCTURE. ---- 89
FIGURE B.8: MODEL (MODEL LIBRARY) LINKED LIST STRUCTURE. ---- 90
FIGURE B.9: LONG (LONG PARAMETERS) DATA STRUCTURE. ---- 90
FIGURE B.10: SHORT (SHORT PARAMETERS) DATA STRUCTURE. ---- 90
FIGURE B.11: TEMP (TEMPERATURE PARAMETERS) DATA STRUCTURE. ---- 90
FIGURE B.12: DIODE (DIODE PARAMETERS) DATA STRUCTURE. ---- 91
FIGURE B.13: JUNCT (JUNCTION PARAMETERS) DATA STRUCTURE. ---- 91
FIGURE B.14: NOISE (NOISE PARAMETERS) DATA STRUCTURE. ---- 91
FIGURE B.15: NODES (NODE DATA) LINKED LIST STRUCTURE. ---- 92
FIGURE B.16: CLIST (CAPACITOR DATA) LINKED LIST STRUCTURE. ---- 93
FIGURE B.17: MLIST (MOSFET DATA) LINKED LIST STRUCTURE. ---- 93
FIGURE B.18: QLIST (TRANSISTOR DATA) LINKED LIST STRUCTURE. ---- 93
FIGURE B.19: RLIST (RESISTOR DATA) LINKED LIST STRUCTURE. ---- 93

FIGURE C.1: POWER PROGRAM MAIN BLOCK DIAGRAM. ---- 94
FIGURE C.2: FLOWCHART FOR POWER MAIN PROGRAM ROUTINE. ---- 95
FIGURE C.3: FLOWCHART FOR "READ_MODELS" SUBROUTINE. ---- 96
FIGURE C.4: FLOWCHART FOR "READ_NETLIST" SUBROUTINE. ---- 97
FIGURE C.5: FLOWCHART FOR "CAP_CALCS" SUBROUTINE. ---- 97
FIGURE C.6: FLOWCHART FOR "COMBINE_SUBCCTS" SUBROUTINE. ---- 98
FIGURE C.7: FLOWCHART FOR "FREE_MEM" SUBROUTINE. ---- 98
FIGURE C.8: FLOWCHART FOR "TIME_NODE_VOLTAGE" SUBROUTINE. ---- 99
FIGURE C.9: FLOWCHART FOR "FILE_CHECK" FUNCTION. ---- 100
FIGURE C. 10: FLOWCHART FOR "LOWER_CASE" SUBROUTINE. ---- 100
FIGURE C. 11: FLOWCHART FOR "READ_MODEL_DATA" SUBROUTINE. ---- 101
FIGURE C. 12: FLOWCHART FOR "PARA_CAP_CALC" SUBROUTINE. ---- 101
FIGURE C.13: FLOWCHART FOR "INTER_CAP_SORT" SUBROUTINE. ---- 102
FIGURE C.14: FLOWCHART FOR "CAP_SORT" SUBROUTINE. ---- 103
FIGURE C.15: FLOWCHART FOR "RES_SORT" SUBROUTINE. ---- 104
FIGURE C.16: FLOWCHART FOR "NODE_CAP" SUBROUTINE. ---- 105
FIGURE C. 17: FLOWCHART FOR "COMBINE_CAP" SUBROUTINE. ---- 106
FIGURE C. 18: FLOWCHART FOR "COMBINE_MOS" SUBROUTINE. ---- 107
FIGURE C.19: FLOWCHART FOR "COMBINE_QUE" SUBROUTINE. ---- 108
FIGURE C.20: FLOWCHART FOR "COMBINE_RES" SUBROUTINE. ---- 109
FIGURE C.21: FLOWCHART FOR "RENAME_REFS" SUBROUTINE. ---- 109
FIGURE C.22: FLOWCHART FOR "ANALYZE" SUBROUTINE. ---- 110
FIGURE C.23: FLOWCHART FOR "POWER_DISS" SUBROUTINE. ---- 111

FIGURE C.24: FLOWCHART FOR "CONVERT" SUBROUTINE. ---------- 112
FIGURE C.25: FLOWCHART FOR "STRTONUM" FUNCTION. ---------- 112
FIGURE C.26: FLOWCHART FOR "VJ_ENTRY" FUNCTION. ---------- 113
FIGURE C.27: FLOWCHART FOR "CALCULATIONS" SUBROUTINE. ---------- 113
FIGURE C.28: FLOWCHART FOR "DATA_FILE_TRANSFER" SUBROUTINE. ---------- 114
FIGURE C.29: FLOWCHART FOR "DRAIN_SOURCE" FUNCTION. ---------- 114

FIGURE D.1: T-CELL XNOR SCHEMATIC DIAGRAM. ---------- 122

List of Symbols

β: The gain factor for transistors.

ε_o: Electric permittivity of free space (vacuum) = 8.854×10^{-14} F/cm

ε_{si}: Electric permittivity of silicon dioxide = $3.9 \cdot \varepsilon_o$

ϕ_b: Bulk potential

γ: Substrate bias effect

AD: Drain Area of a MOSFET

AS: Source Area of a MOSFET

k: Boltzmann's constant = 1.38×10^{-23} J/K

MJ: Bulk p-n grading coefficient

n: emission coefficient

N_i: Carrier concentration

N_A: Carrier density

PB: Built-in bulk p-n potential

PD: Drain Perimeter of a MOSFET

P_d: Dynamic Power Dissipation

PS: Source Perimeter of a MOSFET

P_s: Static Power Dissipation

P_{sc}: Short Circuit Power Dissipation

q: Electronic charge = 1.602×10^{-19} C

T: Temperature in Kelvin

t_{ox}: Oxide thickness

V_{IH}: Input High Voltage

V_{IL}: Input Low Voltage

V_J: Applied reverse bias voltage

V_{OH}: Output High Voltage

V_{OL}: Output Low Voltage

V_t or V_T: The threshold voltage of a MOS transistor.

List of Abbreviations

BiCMOS: Bipolar Complementary Metal Oxide Semiconductor

BJT: Bipolar Junction Transistor

CAD: Computer Aided Design

CMOS: Complementary Metal Oxide Semiconductor

CUT: Circuit Under Test

DUT: Device Under Test

GUI: Graphical User Interface

LSI: Large Scale Integration

MSI: Medium Scale Integration

NMOS: N-Channel Metal Oxide Semiconductor

PMOS: P-Channel Metal Oxide Semiconductor

SSI: Small Scale Integration

TTL: Transistor - Transistor Logic

VLSI: Very Large Scale Integration

List of Nomenclature

Average Power: The average of the powers during various intervals in time.

Average Time: An average of the various times under specific operating conditions.

Duty Cycle: A ratio of the time the output pulse is high, to the total period of the pulse.

Dynamic Power Dissipation: The switching power dissipated by a CMOS circuit.

Fall Time: The time required for a transition from a logic high to a logic low. The time is specified as the period between the 90% and 10% voltage levels.

Interconnect Capacitance: The effective capacitance between the various process layers and devices of a circuit design. The layers can include metal, substrate, polysilicon, and diffusion.

Leakage Current: The current lost in the parasitic diodes of a CMOS circuit.

Parasitic Capacitance: The capacitance intrinsic to a transistor. It includes the gate-drain, gate-source, gate-substrate, drain-substrate, source-substrate, and occasionally drain-source capacitances.

Pattern Dependent Evaluation: An analysis method to examine each node of a circuit, at all time intervals, for a specific set of input test patterns.

Power Dissipation: The total power dissipated or consumed by a CMOS circuit. It generally consists of three parts: Static and Dynamic, Short Circuit.

Rise Time: The time required for a transition from a logic low to a logic high. The time is specified as the period between the 10% and 90% voltage levels.

Short Circuit Current: The current as a result of a short circuit between the supply voltage and ground.

Short Circuit Power Dissipation: The power dissipated as a result of the short circuit current during switching.

Static Power Dissipation: The power dissipated by a CMOS circuit when the circuit is in steady state.

Time Delay (Propagation Delay Time): The time between the specified reference points on the input and output voltage waveforms with the output changing from one defined level to the other defined level.

Total Power: The total power dissipated or consumed over a specified period of time.

Test Vectors: A series of voltages to each input of a circuit under test, with expected, or predictable output results.

1.0 Introduction

Very Large Scale Integration (VLSI) circuit design methods and procedures have changed significantly during the past three decades. The sophistication and reliability of computer aided design tools, simulation tools, and process technology have improved significantly enhancing the design of digital low power circuits.

Power dissipation is a major concern in low power VLSI applications, with transistor densities and clock frequencies increasing, and die sizes decreasing. Performance degradation, elevations in operating temperature and decreased reliability are the results of increased power dissipation in integrated circuits.

A thirst for decreased electronic product sizes by industry and consumers was the main reason for the improvements and changes in design procedures as mentioned above. Examples include the digital and cellular telephones and laptop computers.

Consider the history of the telephone. The rotary telephone existed until the early 1980's. Then a technological boom in the electronics industry brought on significant changes to the development of the cellular and digital telephone, and the demise of the rotary phone. The evolution of cellular and digital phones improved as design and simulation tools improved. The use of complimentary metal oxide semiconductors (CMOS) and bipolar-CMOS (BiCMOS) transistors also had a dramatic affect on the size and operation of these telephones. This created a challenge for designers: to develop integrated circuits, which consume minimal power.

1.1 Objectives

Much of the recent research in VLSI is the estimation of the power dissipation by a device. The primary objective of the research described in this thesis is to explore the requirements needed to develop a low cost educational analysis tool. The ***power*** software program evaluates and analyzes the power dissipation concerns of CMOS circuit designs prior to device fabrication.

A subsequent goal of the research is to determine the validity of computer simulations when compared to other methods of analysis using sample test circuits.

1.2 Justification / Motivation

Researchers use a probabilistic measure of circuit switching activities as the preferred methods of power analysis. This approach uses probabilities to describe all possible logic signals, giving the desired pattern independence during simulations. The problem faced using this method is finding the transition probabilities due to the large scale of most designs.

A thorough literature search at the beginning of this project revealed that little software was available to simulate power dissipation. This has changed over the past six to eight months. Companies such as EPIC Design, Sente, and Mentor Graphics have written commercial estimation and measurement software for power dissipation analysis. These tools use probabilistic analysis methods that are ideal for large-scale designs and for corporations that can afford their prices.

The most simplistic and overlooked method for power dissipation analysis and simulation is pattern dependent evaluation. This method is the main focus of this research. The switching activity of each node is calculated based on user supplied input test vectors. This method is, generally, overlooked because of the dependency of the inputs, however it should not be dismissed as a viable alternative for research purposes and Small Scale to Large Scale Integration (SSI to LSI) CMOS circuit designs.

This research considers other uses for power dissipation analysis, specifically for viewing the dynamic power dissipated by a device for fault detection [27]. It is also ideal for undergraduate and graduate studies, where students should be exposed to all considerations in the design of integrated circuits. The ***power*** simulation program discussed in this thesis will enable the student to appreciate the need for this type of simulation tool, as power dissipation is a major area of concern in integrated circuit design.

1.3 Overview

The following procedure is used to effectively determine the power dissipation in an entire circuit or a single node in the circuit.

- Calculation of total node capacitances, including the parasitic and interconnect capacitances, in the Circuit Under Test (CUT).
- Analysis and calculation of the voltage levels at each node of the CUT. The node voltages depend on the propagation of the user supplied test vectors, throughout the circuit.
- Derivation of a mathematical model to estimate the power dissipation of each node and the total circuit.
- Calculation of individual power dissipation components at each node, for each voltage and time interval. The power dissipation components include the static, dynamic and short-circuit power elements. For this preliminary research the dynamic power is of primary interest. The short-circuit power contribution is of interest when comparing the results to other methods of analysis. The static power is inconsequential at this time due to its significantly smaller contribution.
- Calculation of the estimated and average power dissipation for the CUT.
- Format and storage of all final results in a Matlab formatted text file.
- Combine the Matlab data text file with a Matlab program source code file for a graphical display of the results.

The thesis is divided into three main topic areas. Chapter 2 provides background information about the subject. Chapters 3 and 4 provide the mathematical equations necessary to complete the analysis and the methodology and structure of the software. Chapter 5 demonstrates the software program and compares the results with existing methods. Chapter 6 concludes the thesis and discusses future work to be considered.

2.0 Background

A review of fundamental principles and concepts is necessary prior to discussing this research project. The understanding of power consumption or dissipation in integrated circuits is the first and most important concept to be considered. Simply stated, the total power dissipated in an IC device is the sum of the total static, dynamic and short-circuit powers.

Existing simulation software and tools for the analysis of power dissipation is the next item to be investigated. The method(s) of analysis, mathematical models, the techniques of displaying results, and cost is reviewed as a means of comparison to the proposed research project.

The University of Alberta provides numerous Computer Aided Design (CAD) Software Tools such as Mentor Graphics, Cadence, and Spice. An examination of the properties and methods of circuit extraction is essential to determine the most suitable CAD tool for this project.

The test circuit(s) needed for the purpose of analysis is the final item to be considered. The Circuits Under Test (CUTs) are designed and simulated for research use. These circuits are vital for testing the proposed software program and comparing their results with existing software.

2.1 Power Dissipation

The power dissipation of CMOS devices consists of three components and is governed by the following equation:

$$P = P_s + P_d + P_{sc}, \tag{2.1}$$

where P_s is the static power, P_d is the dynamic power and P_{sc} is the short-circuit power. A simple CMOS inverter, as shown in Figure 2.1, is typically used to define each of the above power terms.

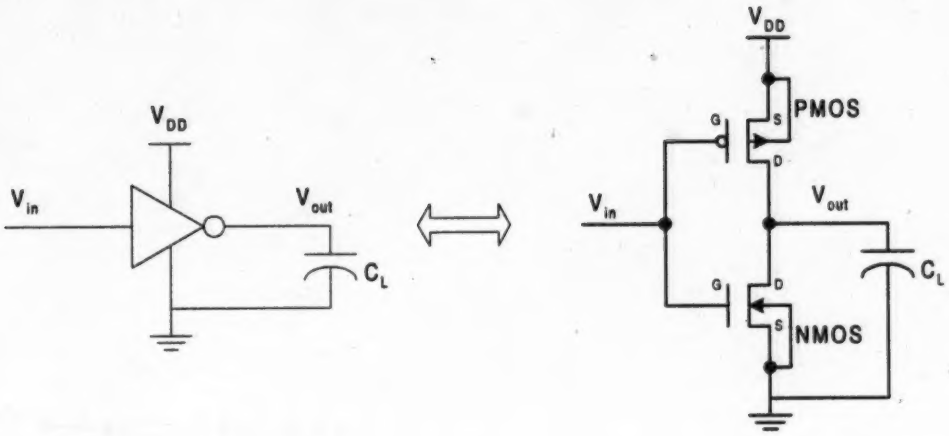


Figure 2.1: Example of a simple CMOS inverter.

The operation of the inverter is as follows: when the input voltage V_{in}, is a high logic level ($\approx V_{DD}$) the PMOS transistor is off and the load capacitor discharges through the NMOS transistor. When the input voltage is a low logic level (≈ 0) the NMOS transistor is off and the capacitor charges to approximately V_{DD}, since the PMOS transistor is on.

2.1.1 Static Power

Leakage currents through non-active transistors and the current drawn from the supply, due to the input voltage of the circuit, are the two causes of the static power, P_s. Static power can be described by

$$P_s = P_{s1} + P_{s2}. \tag{2.2}$$

The term P_{s1} originates from reverse biased leakage currents between diffusion regions and the substrate. Figure 2.2 shows the parasitic diodes of a CMOS inverter, which exist between the n-diffusion wells and the substrate, and the p-diffusion wells and the n-well.

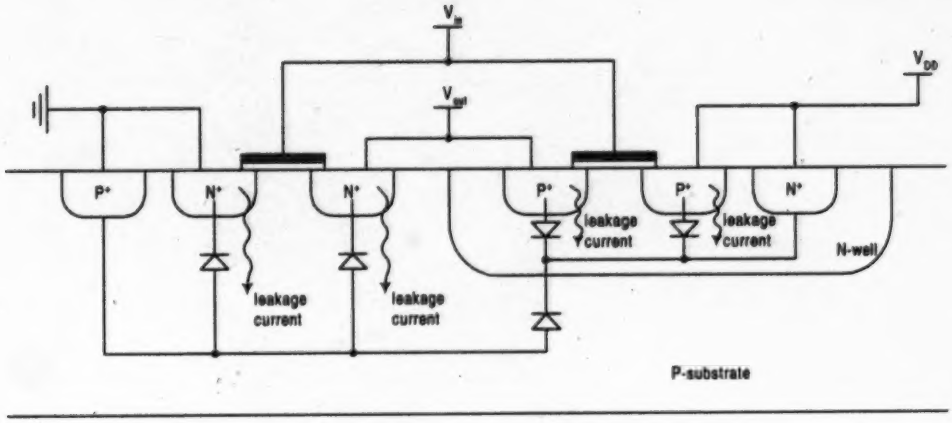


Figure 2.2: Example of a CMOS inverter showing leakage currents.

Since the diodes are reversed biased, only their leakage current contributes to the static power. The leakage current of the diode is given by

$$I_d = I_s\left(e^{\frac{qV}{nkT}} - 1\right), \tag{2.3}$$

where I_s is the reverse saturation current, q is the electronic charge, 1.602×10^{-19} C, V is the diode voltage, n is the emission coefficient of the diode, k is Boltzmann's constant 1.38×10^{-23} J/K, and T is the temperature in Kelvin.

The power dissipation can be calculated using the following equation:

$$P_{s1} = \sum_{i=1}^{n} I_{di} \cdot V_{DD}\,, \tag{2.4}$$

where n represents the number of devices. Note that the total current, I_d, is typically very small, resulting in small power dissipations.

The term P_{s2} is a result of current leaking through transistors that are normally off. When gate biasing is low enough, such that the channel is in weak inversion, current leakage occurs. If the source and drain are too close together then control sharing of the

potential with the gate takes place. If this effect is too strong, a drain voltage dependency of the subthreshold characteristic results. The drain current in this subthreshold region is defined as

$$I_{DSsub} = \frac{W_{eff}}{W_o} \cdot I_o \cdot 10^{(\frac{V_{GS}-V_T}{S})}, \tag{2.5}$$

where W_{eff} is the effective channel width, I_o is the drain current, W_o is the gate width and S is the subthreshold swing parameter. The terms I_o and W_o are used to define V_T and the term S is the voltage swing of the gate needed to decrease the drain current by one decade [1, 19]:

$$I_o = I_o' \cdot (1 - e^{\frac{V_{DS}}{V_t}}) \tag{2.6}$$

$$S \approx 2.3 \cdot V_t \cdot (1 + \frac{C_d}{C_{ox}}) \quad V/decade. \tag{2.7}$$

Equation 2.6 describes the relationship between I_o and V_{DS}. The terms C_d and C_{ox} in Equation 2.7 are the depletion layer capacitance of the source and drain junctions, and the oxide capacitance respectively.

The calculation of P_{s2} requires the worst case leakage current as determined by the worst case threshold voltage

$$V_{T_{worstcase}} = V_T - \Delta V_T, \tag{2.8}$$

where ΔV_T is the change in threshold voltage due to variations and fluctuations in the fabrication process. The power dissipation is then represented by the following equation:

$$P_{s2} = I_{DS_{mean}} \cdot V_{DD}, \tag{2.9}$$

where I_{DSmean} is the mean current value for both the PMOS and NMOS transistors in the inverter.

The total static power dissipation of Equation 2.2 can be rewritten as

$$P_s = \left(\sum_{i=1}^{n} I_{di} + I_{DS\_\_\_} \right) \cdot V_{DD} . \tag{2.10}$$

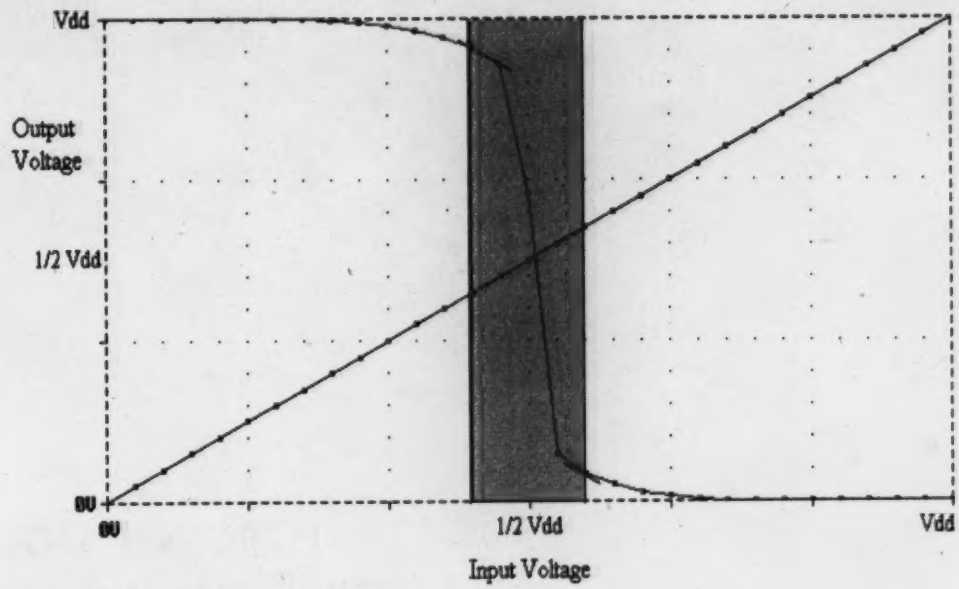
This equation may vary depending on the device technology. In some cases the P_{s1} term is ignored if the leakage current tends to be excessively small, 1 to 10 femtoamps for example. Other cases may ignore the P_{s2} term if the drain and source of the devices have a significantly large separation [5,11,12,13,15,19].

2.1.2 Dynamic Power

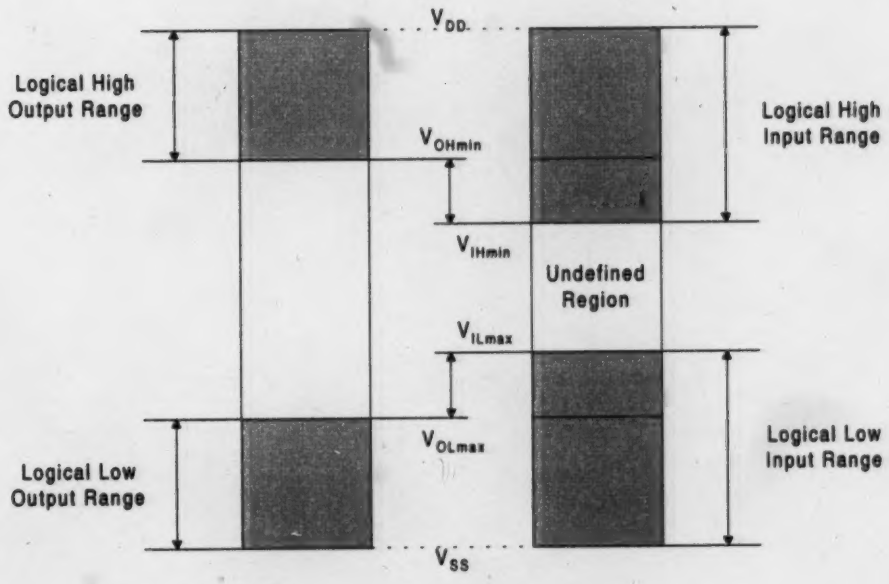
The dynamic power dissipation, P_d from Equation 2.1, is a result of the charging and discharging of the total load capacitance, C_L in Figure 2.4. The short-circuit power dissipation P_{sc} from Equation 2.1, also occurs during this charging and discharging of the load capacitor and is discussed separately in Section 2.1.3.

The transfer characteristic and noise margins of the inverter are shown in Figure 2.3 and are used to aid in the discussion of dynamic power consumption. The non-shaded area of the transfer curve Figure 2.3(a) and the shaded area of the noise margin chart Figure 2.3(b) correspond to the window that dynamic power dissipation may exist.

Two transition states result in the occurrence of the power dissipation by the load capacitance. The first condition is following a change of the input voltage from a level above V_{IHmin} to below $V_{IL,max}$. The second case is during, or following, an input voltage change from below $V_{IL,max}$ to above V_{IHmin}.



(a)



(b)

Figure 2.3: (a) Transfer function and (b) Noise Margin Chart of a CMOS inverter.

To derive the dynamic power of the output load capacitance, a step input to the inverter is assumed. Under ideal operating conditions, this assumption ensures that neither the NMOS or PMOS transistors are on concurrently. The circuits in Figure 2.4 depict changes to the current of the load capacitor C_L, following the step input change.

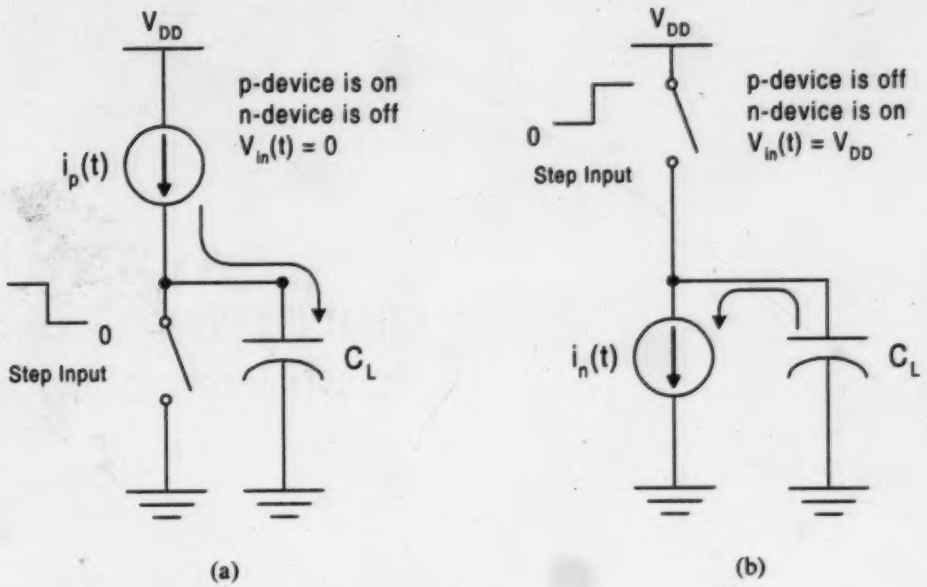


Figure 2.4: CMOS inverter (a) with a logic 1 to 0 step, and (b) with a logic 0 to 1 step.

The effective power with a periodic input waveform is defined by

$$P = \frac{1}{T}\int_0^T i_o(t)v_o(t)dt\,, \tag{2.11}$$

where the term *1/T* is the switching frequency and both the current and voltage are functions of time. For analysis purposes, the switching frequency has a 50% duty cycle. This assumption generally causes the worst case or largest dynamic power dissipation in a CMOS circuit.

The current $i_o(t)$ of the inverter load capacitor C_L is given by

$$i_o = C_L \frac{dv_o}{dt}. \tag{2.12}$$

The term i_o can be positive or negative depending on the current being analyzed, the n-device current $i_n(t)$, or the p-device current $i_p(t)$.

The fundamental equation of the dynamic power dissipation is given as

$$P_d = -\frac{1}{T}\int_0^{T/2} i_n(t)\cdot v_o\, dt + \frac{1}{T}\int_{T/2}^{T} i_p(t)\cdot v_o\, dt, \tag{2.13}$$

where the first term is negative due to C_L discharging to ground, Figure 2.4(b), and the second term is positive due to C_L charging to the supply voltage, Figure 2.4(a). Substitute i_o for the currents in equation 2.13 results in

$$P_d = -\frac{1}{T}\int_0^{T/2} C_L \cdot \frac{dv_o}{dt}\cdot v_o\, dt + \frac{1}{T}\int_{T/2}^{T} C_L \cdot \frac{dv_o}{dt}\cdot v_o\, dt. \tag{2.14}$$

The cancellation of the dt term causes a change in the limits of integration from a time base to a voltage base. The new limits of integration become zero and V_{DD} resulting in

$$P_d = \frac{C_L}{T}\left(\int_0^{V_{DD}} v_o\, dv_o - \int_{V_{DD}}^{0} v_o\, dv_o\right). \tag{2.15}$$

Performing the integrations above results in the final formula for the dynamic power dissipation:

$$P_d = \frac{C_L \cdot V_{DD}^{\,2}}{T} \quad or \quad P_d = \left(C_L \cdot V_{DD}^{\,2}\right)\cdot f. \tag{2.16}$$

2.1.3 Short-circuit Power

The short-circuit power, P_{sc}, occurs at the same time as the dynamic power. This power is independent of load capacitance and is the direct result of both NMOS and PMOS transistors being on simultaneously. The shaded region of the inverter transfer characteristic curve, Figure 2.3(a), and the non-shaded area of the noise margin chart, Figure 2.3(b), visually show the period of time when this power dissipation occurs.

Symmetry of the inverter input is assumed, simplifying the analysis of the short-circuit power dissipation. The time variable, t_{rf}, replaces the rise time and fall time since t_r and t_f are equal. The short-circuit power is then defined as

$$P_{sc} = I_{mean} \cdot V_{DD}. \tag{2.16}$$

Figure 2.5 [2,3,26] is a simple model used to depict the short-circuit current and is useful for the explanation of the mean current

$$I_{mean} = 2 \cdot \left[\frac{1}{T} \int_{t1}^{t2} I(t)dt + \frac{1}{T} \int_{t2}^{t3} I(t)dt \right]. \tag{2.17}$$

Since symmetry is assumed and $I(t)$ is dependent on the size of the transistors, the following relationships are used: $\beta_n = \beta_p = \beta$ and $V_{tn} = -V_{tp} = V_T$, where β is the transistor gain factor, and V_T is the device threshold voltage.

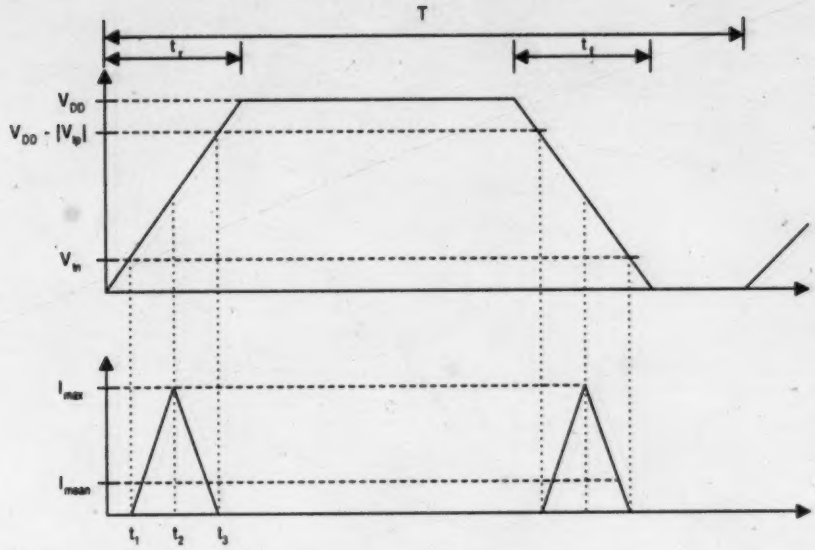


Figure 2.5: Model for short-circuit current and Input switching waveform.

The equation for I_{mean} simplifies to become

$$I_{mean} = 2 \cdot \frac{2}{T} \cdot \left[\int_{t1}^{t2} I(t)dt \right]. \tag{2.18}$$

With the NMOS transistor operating in saturation, the term $I(t)$, in Equation 2.19 is defined by

$$I(t) = \frac{\beta}{2}\left(V_{in}(t) - V_T\right)^2. \tag{2.19}$$

When the input voltage and the times $t1$ and $t2$ are defined as

$$\begin{cases} V_{in}(t) = \dfrac{V_{DD}}{t_{rf}} \cdot t & t_1 = \dfrac{V_t}{V_{DD}} \cdot t_{rf} \qquad t_2 = \dfrac{t_{rf}}{2}, \end{cases} \tag{2.20}$$

the integral component of I_{mean} can be solved and reduced when substituting these equations and Equation 2.19 into Equation 2.18.

The short-circuit power dissipation can be rewritten when I_{mean} is replaced by its equivalent. Therefore P_{sc} becomes

$$P_{sc} = \frac{\beta}{12} \cdot \frac{t_{rf}}{T} \cdot (V_{DD} - 2 \cdot V_T)^3, \tag{2.20}$$

where T is the period of the input signal.

It is important to note that this derivation is under no-load conditions and the short-circuit current depends on β, the input rise time and the fall time. Changes in the load capacitance also effect the significance of the short-circuit power. Thus, the dynamic power becomes the dominant contributor in the total power consumption of a CMOS device. The short-circuit dissipation will be a fraction (< 20%) of the total power with equal rise and fall times of the input and output [1,2,3,26].

In general terms, the power dissipation caused by a short-circuit can be stated as

$$P_{sc} = \frac{t_f + t_r}{2 \cdot T} \cdot (V_{DD} \cdot I_{mean}), \tag{2.21}$$

where T is the period of the input signal.

2.1.4 Total Power Dissipation

The total power dissipation is the sum of the static power, Equation 2.10, the dynamic power, Equation 2.15, and the short-circuit power, Equation 2.21. Substituting these equations into Equation 2.1, the total power can be written as

$$P = \left[\left(\sum_{i=1}^{n} I_{di} + I_{DS_{mean}} \right) \cdot V_{DD} \right] + \left[\frac{C_L \cdot V_{DD}^2}{T} \right] + \left[\frac{t_f + t_r}{2 \cdot T} \cdot (V_{DD} \cdot I_{mean}) \right], \tag{2.22}$$

where T is the period of the input switching signal.

In large, complex circuits it is more practical to approximate the total power dissipation. The dynamic power of Equation 2.16 is generally the dominant cause of power dissipation in CMOS circuits and can be modified from its original form [26].

The new equation for the dynamic power becomes

$$P_{d_{approx}} = \sum_{tr=0}^{max\ transitions} \left(\sum_{n=1}^{max\ nodes} \frac{C_n \cdot \Delta V_n^{\,2}}{t_{tr,n}} \right). \tag{2.23}$$

The term C_n, which represents the total node capacitance, replaces C_L in the original equation and ΔV_n replaces V_{DD} as the total change in node voltage. The total period T is replaced by the time $t_{tr,n}$, since the input signal may not be periodic. Thus the power dissipated at a node is calculated during each transition of the gate voltage.

2.2 Existing Power Analysis Software

Power dissipation analysis methods have been researched extensively during the last decade. The availability of commercial power analysis tools has, only recently, started to increase. These tools enable designers to generate device designs and test them for effects of power consumption before fabrication. PowerMill, Design Power, Lsim Power Analyst, PowerCalc, and XPOWER are some power analysis tools available [20,21,22].

2.2.1 Description of Existing Software

Each software tool performs similar functions, however any may be tailored for different applications. Some tools have advantages that make them better than the rest and disadvantages which makes them secondary to the others.

PowerMill, by Epic, can simulate design blocks and complete chips. The instantaneous, average, RMS, and power rail currents of the design are easily and accurately analyzed. As well, the use of simplified, table driven device models and improved analysis models increase the speed of the simulation. The addition of hazard and glitch contributions, the ability to identify "hot spots", and the ability to read netlist, vector and technology files, improve the selling points of this software [23,24].

Synopsis' Design Power, uses simulation based analysis for power estimation. It is a single, complete integrated environment for analysis in many design stages. Sequential designs, multiple and gated clock designs, and hierarchical designs are well suited for this software. Internal cell and leakage power analysis is also supported for this design software. The ability to link directly to Synopsis for simulation data is yet another plus [20].

Lsim Power Analyst, by Mentor Graphics, allows for large designs in a flat or top-down hierarchy. It calculates and analyzes ground bounce, and electro-migration powers, in addition to total power consumption. Transistor level analysis is an additional advantage of this program [20].

PowerCalc, by COMPASS Design Automation, computes the average power dissipated by wiring and device instances. The estimation of the power dissipated by current glitch transitions is also calculated [21]. Standard cell and libraries are included to improve the speed and accuracy of the analysis.

Finally, XPOWER, a research designed software, analyzes and graphs dynamic power consumption. It is used primarily in research applications, for the validation of asynchronous digital circuit [22]. Section 2.2.2 discusses this software in more detail.

2.2.2 Thesis Software Program vs. Existing Software.

Probabilistic, statistical, or pattern independent analysis of circuit switching activities, is the preferred analysis method used by researchers and designers. Many of the previously mentioned software tools use this method of analysis for power estimation. One of the problems using this analysis method is finding transition probabilities in large scaled designs.

Pattern-dependant evaluation is the most simplistic and overlooked method of power dissipation analysis and simulation. The switching activity of each node is

calculated based on user supplied input test vectors. It is an ideal approach for SSI, MSI, and LSI CMOS designs.

XPOWER is the most similar of the existing software programs investigated. When compared to this thesis research only the dynamic power dissipation analysis is examined. The software is simplistic and provides a general estimation of the total consumption of power by a device. The program differs from this research in that generic libraries are used in conjunction with a Spice only format, and the short-circuit power contribution is examined.

The ***power*** program calculates the total node energy consumed at any given time, the total power dissipated and the average power dissipated. The program uses Cadence model libraries, but is capable of using generic cell libraries. The netlist is in Spice format, as are the majority of analysis tools, however the netlist can be generated from schematic or device layout design.

PowerMill is the most sophisticated of the other software tools. The block diagram, in figure 2.6 [24], describes the basic components of PowerMill. Complexity and price are the drawbacks of using this software in University applications.

The research outlined and described in this thesis provides the necessary power analysis requirements without the exorbitant cost. It is not as sophisticated as PowerMill, refer to Figure 2.7, however the analysis is sufficient to obtain initial power dissipation results.

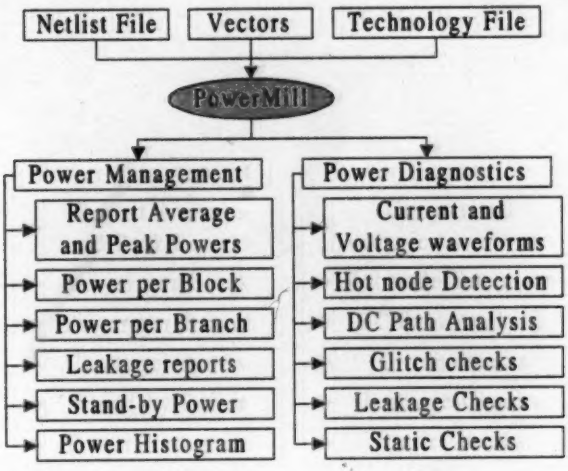


Figure 2.6: Procedural block diagram of PowerMill software program.

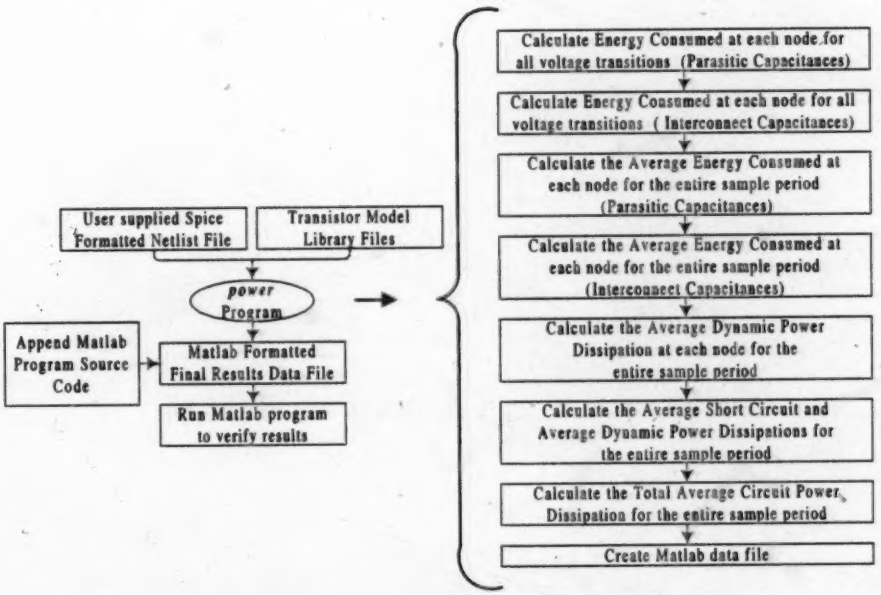

User supplied Spice Formatted Netlist File
Transistor Model Library Files
power Program
Append Matlab Program Source Code
Matlab Formatted Final Results Data File
Run Matlab program to verify results
Calculate Energy Consumed at each node for all voltage transitions (Parasitic Capacitances)
Calculate Energy Consumed at each node for all voltage transitions (Interconnect Capacitances)
Calculate the Average Energy Consumed at each node for the entire sample period (Parasitic Capacitances)
Calculate the Average Energy Consumed at each node for the entire sample period (Interconnect Capacitances)
Calculate the Average Dynamic Power Dissipation at each node for the entire sample period
Calculate the Average Short Circuit and Average Dynamic Power Dissipations for the entire sample period
Calculate the Total Average Circuit Power Dissipation for the entire sample period
Create Matlab data file

Figure 2.7: Procedural block diagram of thesis software program.

2.3 CAD Software Components

Computer Aided Design (CAD) tools are essential in any digital circuit design. Software programs such as Spice, Cadence, Mentor graphics, and Magic are all examples of such tools. Each CAD package has unique and possibly similar traits and capabilities. For example, Cadence, Mentor Graphics and Spice all have schematic entry and simulation abilities, in various degrees, whereas Magic does not. Device layout of a design can be implemented in Magic and Cadence, but not in Spice and Mentor Graphics. Since Cadence encompasses all necessary capabilities, and it is supported at the University of Alberta, it was selected as the primary design tool for this research.

A general procedure needed to create a functional circuit, once a design has been formulated and conceptualized, is as follows:

1. Schematic entry into a CAD tool.
2. Verification of CAD design rules.
3. Simulation of design, ensuring correct circuit operation.
4. Repeat steps 1→3 for any corrections, then proceed to Step 5.
5. Device layout of the design. (i.e. transistor level)
6. Simulation of the design, ensuring correct circuit operation.
7. Simulation of design to determine layout, power consumption or timing problems.
8. Repeat steps 5→7 for any corrections, then proceed to Step 9.
9. Submit design for fabrication.

In some cases, steps 1→4 are omitted and the design is entered entirely in device layout format. This is typical for small designs to simplify the amount of work required.

Once the design has been entered as in steps 1 and/or 5, simulation of the circuit is required to ensure correct circuit operation. This is accomplished by supplying voltages to the inputs of the circuit, also known as providing test vector inputs. Once verification is complete, a netlist representation of the circuit, voltage supplies included, can be exported. This is one of the files necessary for the research program.

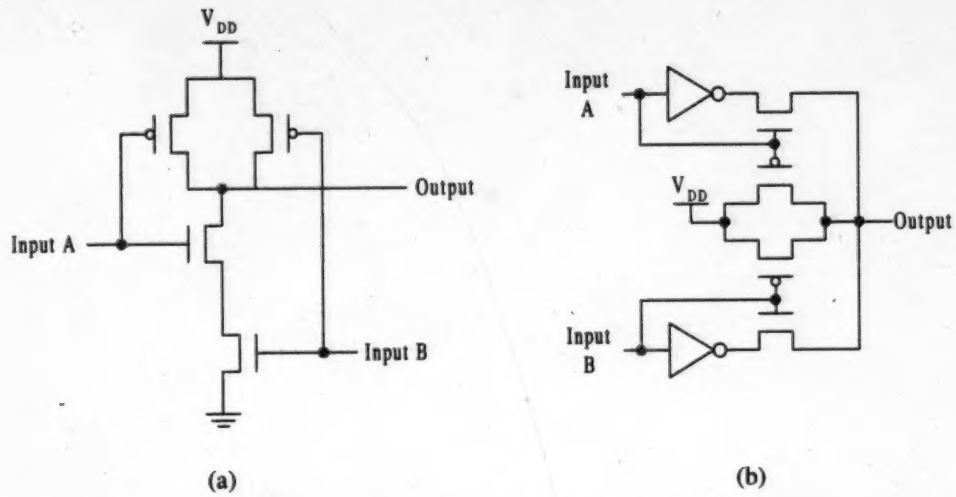
The remaining files needed for the research program are the library or technology files. There are files for NMOS, PMOS, and BJT transistor model technologies, and are supplied to the University of Alberta by the Canadian Microelectronics Corporation. It should be noted that the technology files can be used with netlists generated by other, CAD tools, with some minor modifications. Sample technology library files, used for this program, are included in Appendix A.

2.4 Circuit Under Test (CUT)

Test cases are a necessity to help develop and test the functionality of this thesis software. Digital logic gate designs and fully functional circuits are used as the test cases to verify the program results. These results are then compared with other methods of analysis to determine the software accuracy.

The six most common logic gates, including AND, NAND, OR, NOR, XOR, and XNOR were used in CMOS configurations. General library formats for the various gates include four to ten NMOS and PMOS transistors. The simplest designs are used to assist in the various stages of software development.

An important caution: all test circuits must have gate layouts as the highest level of the hierarchy. This is essential for the software to operate correctly, and allows various transistor representations of the gates to be accounted and tested for. Figure 2.8 represents two forms of a CMOS NAND gate [25]. A standard NAND gate is shown in Figure 2.8(a) and a double-pass transistor NAND gate is shown in Figure 2.8(b). If the two circuits are not represented in gate format, within the netlist, the software will not recognize their configuration differences.



(a) (b)

Figure 2.8: (a) A simple CMOS NAND gate, and (b) A double-pass CMOS NAND gate.

Two additional, fully functional circuits were also exhaustively tested to verify the correct operation of the software. The first was a single bit adder and the second was a 4-2 compressor circuit, Figure 2.9 [25]. The netlist for the 4-2 compressor circuit is provided in Appendix D3.

The adder and compressor were chosen as the primary test circuits since they use various types of gate configurations. Therefore standard and user-defined gate libraries can be used to describe the circuit netlist.

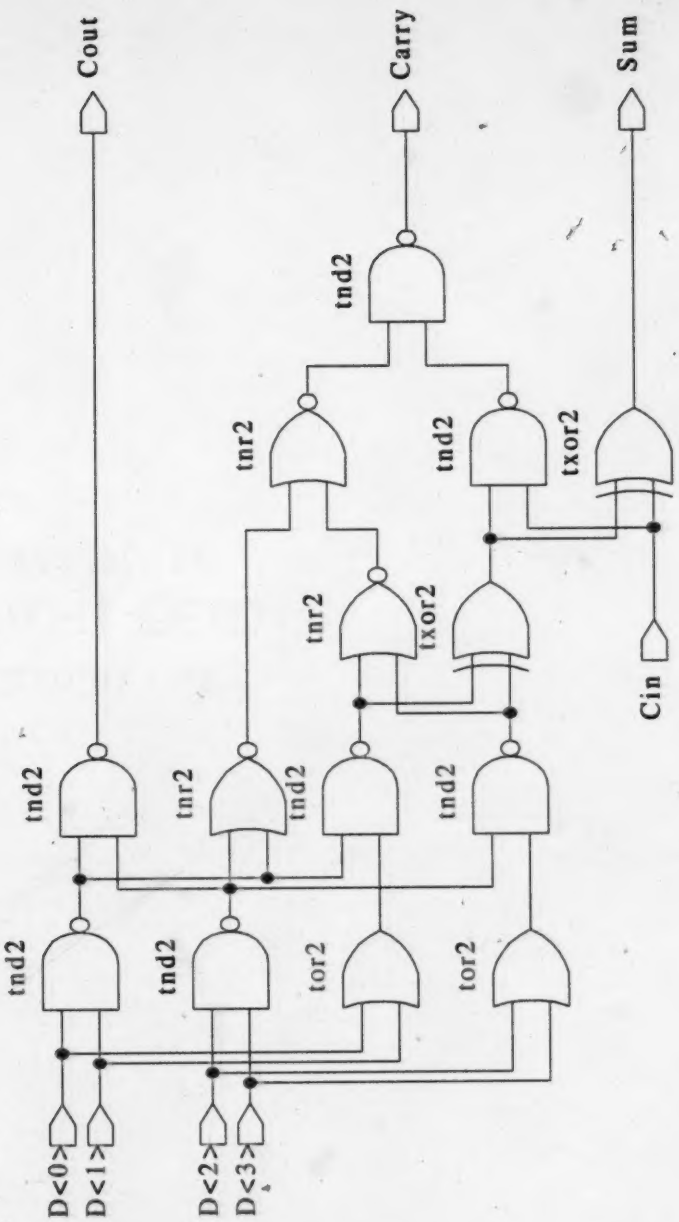


Figure 2.9: A 4→2 Compressor Circuit.

3.0 Models and Analysis Techniques

The issue of pattern dependency vs. probabilistic analysis methods, in conjunction with the mathematical models available and analysis methods used, must be addressed prior to discussing the software generated for this thesis.

3.1 Mathematical Analysis Methods and Models

Research into the theory of low power CMOS design revolves around modeling power consumption. The common approach to this analysis is to estimate the power consumption via probabilistic methods as opposed to pattern dependent analysis. The comparisons of these methods and the models used for this research are presented in further detail.

3.1.1 Pattern Dependency Vs Probabilistic Analysis Methods

Probability or statistical methods compute the fraction of cycles in which an input signal, or test vector, makes a transition. The transition can be from logic 0 to logic 1, or visa-versa, [18]. This method is considered to be weakly pattern dependent and is ideal for large designs and combinational circuit designs. The difficulty with this method is computing the transition probabilities, especially with feedback circuit elements (latches, for example).

The dynamic power definition of Equation 2.16 is rewritten as Equation 3.1 to account for the probability analysis method.

$$P_d = \frac{\alpha \cdot C_{TOT} \cdot V_{DD}^{\,2}}{t_p}. \tag{3.1}$$

The term C_{TOT} replaces C_L in the original equation and represents the total capacitance driven by the gate outputs in the circuit. The term α represents the switching probability, or activity ratio, and can be defined as the estimated percent activity of the node, functioning at the maximum clock frequency, $f_{clk} = 1/t_p$.

In contrast to the weakly pattern dependent method of analysis, this thesis focuses on strongly pattern dependent estimation of power dissipation. Strong pattern dependency provides a more detailed description of power dissipation at each node in the circuit design. It is ideal for small and medium designs where the input patterns are known. As well it can analyze node power consumed at specific intervals of time, for certain input test vectors. Conversely, the simplistic method may require an exhaustive number of test vectors to verify all possible input combinations. This is a moot point when detailed studies are required for research.

3.1.2 Mathematical Models

This thesis research depends on various mathematical equations required for circuit analysis. Many of these models are standard in determining parasitic and interconnect capacitances, node voltages, and the various contributions to power dissipation. Each of these is discussed in further detail.

3.1.2.1 Capacitance Models

The total gate capacitance C_g of a MOSFET is defined as the sum of the intrinsic and extrinsic gate capacitances and is represented by

$$C_g = C_{g_{\text{intrinsic}}} + C_{g_{\text{extrinsic}}}. \tag{3.2}$$

The intrinsic capacitance is defined as

$$C_{g_{\text{intrinsic}}} = Length \cdot Width \cdot C_{ox}, \tag{3.3}$$

where the length and width are defined by the technology or by the user. The oxide capacitance, C_{ox}, is defined by the properties of silicon dioxide, and is represented by

$$C_{ox} = \frac{\varepsilon_0 \cdot \varepsilon_{SiO_2}}{t_{ox}}, \tag{3.4}$$

where t_{ox}, is the oxide thickness and is supplied by the technology library files, ε_0 is the electrical permittivity of free space (vacuum) with a constant of 8.854 x 10^{-14} F/cm, and ε_{SiO_2} is the relative permittivity of silicon dioxide with a constant of 3.9.

The extrinsic capacitance is defined as

$$C_{g_{\text{extrinsic}}} = \left(Width \cdot C_{gso}\right) + \left(Width \cdot C_{gdo}\right) + \left(2 \cdot Length \cdot C_{gbo}\right), \tag{3.5}$$

where C_{gbo} is the gate-bulk overlap capacitance per unit channel length, C_{gdo} is the gate-drain overlap capacitance per unit channel width, and C_{gso} is the gate-source overlap capacitance per unit channel width. The technology library files supply these parameters.

The drain or source capacitance is represented by

$$C_d \text{ or } C_s = C_{Area} + C_{Perimeter}. \tag{3.6}$$

The first parameter, C_{Area}, is the area contribution to the total capacitance and the other parameter, $C_{Perimeter}$ is the perimeter contribution to the total capacitance.

The capacitance from the area contribution is given by

$$C_{Area} = Area \cdot CJ \cdot \left(1 + \frac{VJ}{PB}\right)^{-MJ}, \tag{3.7}$$

where the *Area* term is defined by AD or AS (source or drain area) from the netlist file. The term *VJ* is the magnitude of the applied reverse bias voltage. The remaining terms are provided by the model library files where, *CJ* is defined as the zero-bias capacitance per junction area, *PB* is defined as the built-in bulk p-n potential, and *MJ* is the bulk p-n grading coefficient of the junction bottom.

The perimeter contribution is defined as

$$C_{Perimeter} = Perimeter \cdot CJSW \cdot \left(1 + \frac{VJ}{PBSW}\right)^{-MJSW} \quad (3.8)$$

where the *Perimeter* term is defined by PD or PS (source or drain perimeter) from the netlist file. The remaining terms are provided by the model library files where, *CJSW* is defined as the zero-bias-junction capacitance per junction periphery, *PBSW* is defined as the built-in bulk p-n potential, and *MJSW* is defined as the bulk p-n grading coefficient of the junction sidewall.

3.1.2.2 Node Voltage Models

CMOS gates use various configurations of NMOS and PMOS transistors. The test gates and circuits used in this thesis are connected in a standard configuration as shown in Figure 3.1(a).

A PMOS transistor passes a strong logic '1' when $V_{in} < V_{tp} + V_{DD}$ and the source (drain) node is connected to the supply. In this case, the load capacitor will charge to the supply voltage V_{DD}. The same transistor passes a weak logic '0' when the source is connected to ground as shown in Figure 3.1(b). In this case, the capacitor voltage will discharge to approximately $|V_{tp}|$, which is not desirable.

In contrast, a NMOS transistor passes a strong logic '0' when $V_{in} > V_{tn}$ and the source (drain) node is connected to ground. In this case, the load capacitance will discharge to ground or zero volts. The same transistor will pass a weak logic '1' when the source is connected to the supply. Thus the capacitor will charge to $V_{DD} - V_{tn}$.

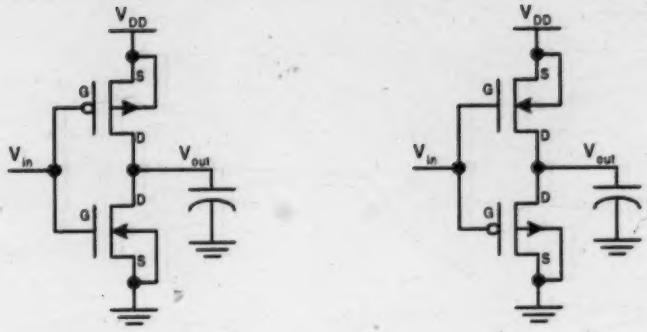


Figure 3.1: (a) Standard CMOS Configuration, and (b) Non-Standard CMOS Configuration.

A second consideration is the definition of the threshold voltage, V_t. The resulting V_t depends on the configuration of the MOSFETs as discussed above and how the substrate is connected. The principal equation for V_t is defined as

$$V_t = V_{to} + \gamma \cdot \left(\sqrt{2 \cdot \phi_b + |V_{sb}|} - \sqrt{2 \cdot \phi_b} \right), \tag{3.9}$$

where V_{to} is the threshold voltage with $V_{sb} = 0$. The substrate bias voltage, V_{sb}, is the difference between the source and the substrate voltages. The term ϕ_b represents the bulk potential, which describes the doping of the substrate. Equation 3.10 defines ϕ_b as

$$\phi_b = \frac{k \cdot T}{q} \ln\left(\frac{N_A}{N_i} \right), \tag{3.10}$$

where k is Boltzmann's constant, T is temperature in Kelvin, q is the electronic charge, N_A is the carrier density, and N_i is the carrier concentration. The constant gamma, γ, is described as the substrate bias effect and is expressed as

$$\gamma = \frac{t_{ox}}{\varepsilon_{ox}} \cdot \sqrt{2 \cdot q \cdot \varepsilon_{si} \cdot N_A}, \tag{3.11}$$

where t_{ox} is the oxide thickness, ε_{ox} is the dielectric constant of silicon dioxide and ε_{si} is the dielectric constant of the silicon substrate.

The standard configuration connects the source and substrate of the PMOS transistor(s) to the most positive potential. The NMOS transistor(s) connect the source and substrate to ground potential. The substrate bias voltage, V_{sb}, is approximately zero for all cases with these characteristics. Thus the second term in Equation 3.9 can be omitted resulting in the following equation:

$$V_t = V_{to}. \tag{3.12}$$

This simplified threshold voltage is used for the remainder of the analysis.

The effects of the threshold voltage on the two configurations are explained using the examples in Figure 3.1. The voltage at the load capacitor in Figures 3.1(a) will be V_{DD} or ground respectively when a strong logic '1' or logic '0' is passed. Conversely, when a weak logic '1' or logic '0' is passed the voltage at the load capacitor in Figures 3.1(b) will be $V_{DD} - V_t$, or V_t respectively.

3.1.2.3 Power Dissipation Models

The total power consumption by a circuit is the sum of the static, dynamic, and short-circuit dissipations as described by Equation 2.1. The calculations used for this thesis omit the effects of the static dissipations, since they tend to be significantly less than the dynamic and short-circuit powers [2,3,5,26]. The short-circuit power is typically less than 20% of the total power dissipated [2,3,5,26], and is considered only for comparisons with existing software. Therefore, Equation 2.24 is used as a basis for the remainder of the power dissipation analysis.

The dynamic energy consumption is analyzed before the calculations of the average power dissipated at each node. The energy is divided into individual elements as shown in Equation 3.13, which defines the contributions of the dynamic energy at any given node, for any given transition.

$$E_{dyn\_tot} = E_{dyn\_interconnect} + E_{dyn\_parasitic} \tag{3.13}$$

$E_{dyn\_interconnect}$ represents the energy consumption due to interconnect capacitances and $E_{dyn\_parasitic}$ represents the energy consumption due to MOSFET parasitic capacitances. Each component is calculated using the same formula,

$$E_{dyn} = C_X \cdot (\Delta V)^2, \tag{3.14}$$

where C_X is either the parasitic or interconnect capacitance(s). The ΔV term is determined by the change of voltage from the previous time internal to the current interval, as defined by equation 3.15.

$$\Delta V = V\{Current\,Time\,Interval\} - V\{Last\,Time\,Inverval\} \tag{3.15}$$

The method of calculating the energy contribution from the parasitic capacitance is as follows:

1. Set all initial energy contributions to zero.
2. Determine if the result for ΔV is for an input or internal node.
3. If the result is for an internal node, the energy consumption determined by Equation 3.14 is divided by 2, (the transistor consumes ½ of the energy during switching). If the result is for an input node, the consumed energy is the entire energy calculated by Equation 3.14, (the capacitor consumes all of the energy).

The method of calculating the energy contribution from the interconnect capacitances is as follows:

1. Set all initial energy contributions to zero.
2. Find the two nodes for the interconnect capacitance under test.
3. Determine the result for ΔV on one node of the capacitor, NodeA.
4. If the result is negative, then multiply by negative 1.
5. Determine the result for ΔV on the other node of the capacitor, NodeB.

6. If the result is negative, then multiply by negative 1.

7. Determine the final ΔV by subtracting NodeB voltage from NodeA voltage.

8. a) If the result is positive: calculate the energy contribution using Equation 3.14 and add the result to existing energies for NodeB.

   b) If the result is negative: Multiply ΔV by negative one, then calculate the energy contribution using Equation 3.14 and add the result to existing energies for NodeA.

The dynamic energy components at each node are totaled for all time intervals. The average dynamic energy is then calculated by dividing the final dynamic energy result by the total number of switching transitions, either logic '1' to logic '0' or visa-versa as shown by Equation 3.16.

$$E_{ave} = \frac{E_{dyn}}{Total\ Number\ of\ Single\ Transitions} \tag{3.16}$$

The same approach is applied when determining the average interconnect contribution.

The average power dissipation of a node from the parasitic capacitances is calculated using Equation 3.17. This average power is dissipated during the entire test period T.

$$P_{ave} = \frac{E_{ave}}{T} \tag{3.17}$$

Equation 3.17 is also used to calculate the average power dissipation at a node due to the contribution of the interconnect capacitance.

The sum of the parasitic and interconnect node dissipations during the test period results in the total dynamic power of a node. Therefore, the total dynamic power consumed by the circuit is determined by

$$P_{ave\_dyn} = \frac{\sum_{all\ nodes} P_{ave}}{Number\ of\ Nodes}. \tag{3.18}$$

The short-circuit dissipation is determined by

$$P_{Short-Circuit} = \left(\frac{average\ time}{switching\ time} \right) \cdot V_{DD} \cdot I_{Peak}, \tag{3.19}$$

where I_{peak} is the average of the largest peak currents for each node. The *switching time* is the period at which the inputs change, and the average time is defined by

$$average\ time = \frac{t_r + t_f}{2}, \tag{3.20}$$

where t_r is the average of all input rise times and t_f is the average of all input fall times.

The total average power dissipation of the circuit is then defined as the sum of the average dynamic power and the average short-circuit power dissipations during the entire test pattern period.

3.2 Circuit Analysis Techniques and Discussion

A flagging approach was used to examine each MOSFET drain, gate and source nodes of the circuit:

- A *zero* flag represents an unchecked node.
- A *one* flag represents a visited node that has yet to be completely analyzed.
- A *two* flag represents a node that is connected directly to a voltage supply input.
- A *three* flag represents a visited node that results in a non-changing voltage level.

The analysis begins with a loop counter set for a maximum of three. This is designed so that all node elements of the circuit will be evaluated and remain at the steady-state value.

For each pass through the loop all transistors are examined. Figure 3.2 depicts the overall loop structure. The type of transistor (NMOS or PMOS) and the state of the gate flag and voltage will determine the remainder of the analysis for the circuit with respect to the transistor under test.

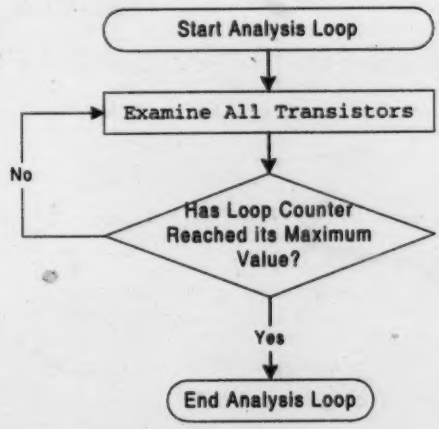


Figure 3.2: Block diagram of overall analysis loop.

As previously stated, three criteria are required for the evaluation of the transistor state of operation. The first is the state of the gate flag (see Figure 3.3). Although four paths exist only the *one* and *two* flag states are checked.

The second criterion is that the transistor gate voltage will match one of two possible ranges. The first voltage range is greater than or equal to zero and less than half the supply voltage ($0 \leq V_1 < V_{DD}/2$), where $V_{DD}/2$ is the typical switching point of the input to the transistor. The other range is greater than or equal to half of the supply voltage and less than or equal to the supply ($V_{DD}/2 \leq V_2 \leq V_{DD}$).

The third criterion is the model type of the transistor under test. PMOS or NMOS are the two possibilities in CMOS circuits. The transistor type along with the voltage level will determine the correct analysis path for the transistor under test (see Figure 3.3).

Once the path has been established the remainder of the circuit analysis can be completed with respect to the transistor under test. Included in the chosen path are the settings of the drain, gate and source flags.

Two of the four possible paths will cause the remainder of the circuit to be analyzed, with respect to the transistor under test. These are paths A and C. The two remaining paths will test the transistor only by changing the appropriate drain and source voltages, and the drain, gate, and source flags depending on the state of the transistor.

If path A or C is chosen, one of the drain or source node voltages will change and the drain, gate, and source flags will change as necessary. Once these changes have been completed, a routine is called. The parameters passed to the routine include the current transistor under test, the current node under test, the current flag value for the node under test, and the current voltage level of the node under test. This routine is discussed further in the next section.

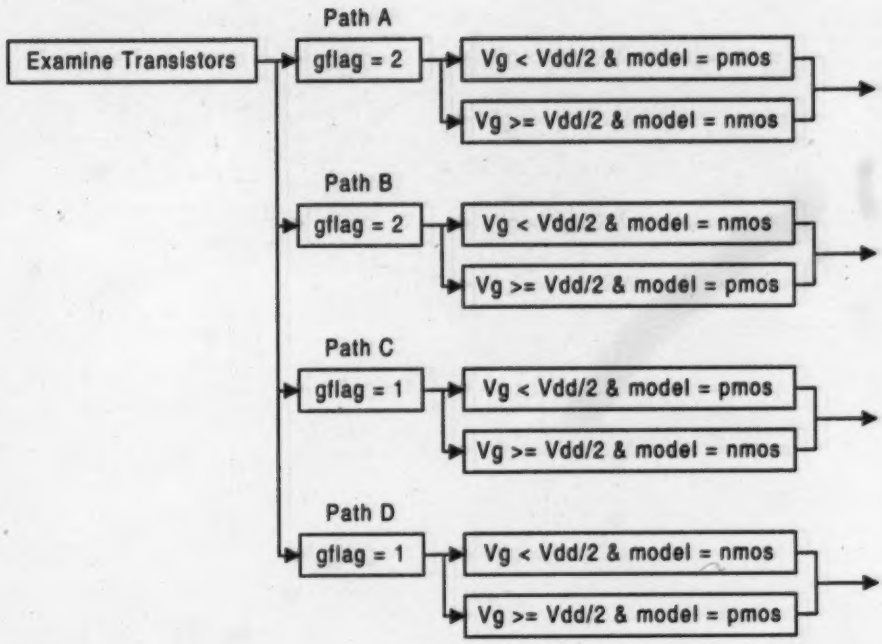


Figure 3.3: Block diagram of internal node analysis from "time_node_voltage" subroutine.

The remainder of the circuit under test is examined when the subroutine is called. The parameters passed are used to compare current settings of transistor node flags and voltages, and allow for corrections. Under certain circumstances, the voltage of the current transistor under test and its corresponding node under test will change to reflect current circuit parameters.

4.0 Subroutine Description and Software Application

The ***power*** software program was written in "C" to take advantage of the UNIX operating system, and existing software that is also written in "C". Cadence and other CAD tools operate under a UNIX platform; therefore, this program was no exception. It was designed to compile and execute on platforms including, SUN-OS, Solaris, and HP-UX.

The software structure, methods, and procedures for the power analysis program are discussed in the remainder of this section. Appendix C.1 details a block flow representation of the program. Some of the modules require the use of one or two header files. Appendices B.1 and B.2 list these header file descriptions.

4.1 "power" Main Program Routine

power calculates node interconnect and parasitic capacitances and power dissipations of logic circuits in a design stage. A netlist file from Cadence or another design tool is required in the working directory in which the program is executed. The technology files, supplied by the Canadian Microelectronics Corporation, accompany the netlist file. These include the NMOS, PMOS, and NPN/PNP transistor libraries. The program is executed by entering the following instruction at the command line:

power [netlist file name] [output file name]

The main program routine "power" has three specific functions as outlined in Figure 4.1. A flowchart for the program scheme is provided in Appendix C.1.

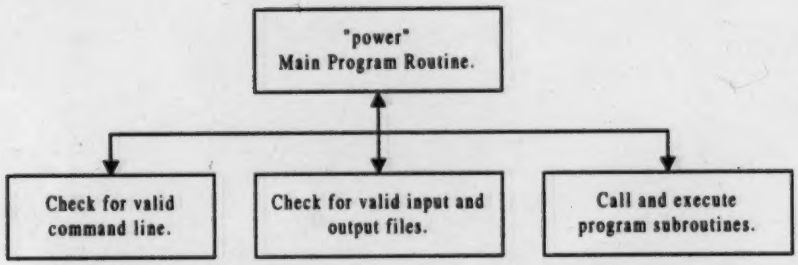

"power"
Main Program Routine.
Check for valid command line.
Check for valid input and output files.
Call and execute program subroutines.

Figure 4.1: Block diagram of main program "power" routine.

The first function is to validate the command line entered by the user. There should be three arguments with the first being the program name. The second argument is the input netlist file name followed by the third argument, the output file name. If there are not exactly three variables, a prompt will be displayed showing the correct usage of the program. If the correct number of entries exist, the input and the output files are tested for validity.

The second function of the main routine is to check the integrity of the input and output file names. If the input file does not exist or does not contain correct data an error message is displayed. This message states that an invalid or empty netlist file is present and prompts the user to check the file. If the output file specified by the user currently exists, an error message is displayed, stating that the output file exists. A prompt tells the user to delete, rename, or specify a new file name. If any of these situations occurs, the program will terminate and the user must re-enter the command line. If the two files are valid, the input file is opened for reading and the output file is opened for writing.

The third and final operation of the main routine is to call the rest of the program subroutines (see Figure 4.2). These will be referred to as level-one subroutines. When all calls have been successfully executed, the input and output files are closed and the program ends.

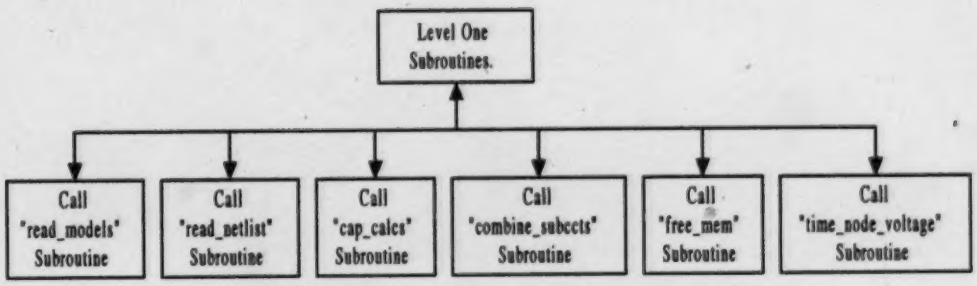


Figure 4.2: Level-one program subroutines.

4.2 Level-one Subroutines and Functions

This section discusses all level-one subroutine and function calls from the main program routine. The level-one subroutine flowcharts are provided in Appendix C.2.

4.2.1 "read_models" Subroutine

The "read_models" subroutine has three specific objectives as detailed in Figure 4.3. Three level-two subroutines or functions are required to meet each of the objectives of this subroutine.

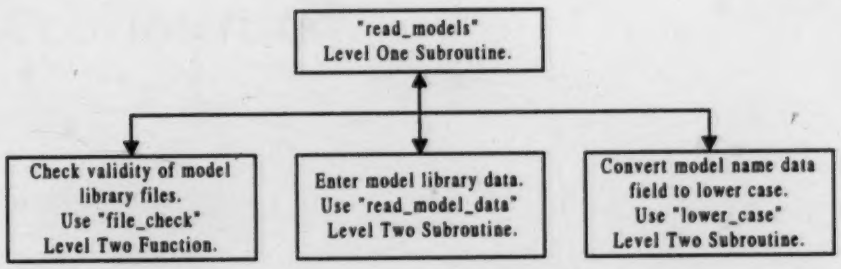


Figure 4.3: Block diagram of "read_models" subroutine.

The first priority of this subroutine is to check the integrity of the model library files, NMOSLIB and PMOSLIB. This is accomplished by calling the "file_check" level-two function. If either of the library files is non-existent or not in the current working

directory, an error message is displayed stating that the model library file is invalid. If either of the library files exist without any data, an error message will also be displayed stating the above error message. The program will terminate following the error message and the user must correct the fault before re-entering the command line.

The second goal is to enter the model data required for analysis calculations, once the validity of the library files is accepted. This is accomplished by calling the "read_model_data" level 2 subroutine. The first set of data is entered from the NMOS library file, followed by the PMOS library data. The transistor data is not entered at this stage of the program design, but will be considered for later revisions. The data entered is converted from string format to double format.

The final goal of this sub-program file is to convert the name field from upper case ASCII to lower case ASCII. This is accomplished by calling the "lower_case" level-two subroutine. Comparisons with the netlist data in a later part of the program, is the reason for this conversion. Control is returned to the main program routine, once the subroutine has successfully completed its tasks.

4.2.2 "read_netlist" Subroutine

The primary objective of the "read_netlist" subroutine is detailed in Figure 4.4. This subroutine reads and sorts the data from the user-supplied circuit netlist file, into linked lists. The specific information contained in the data file includes: sub-circuits, capacitances, resistances, MOSFET and BJT transistors, and voltage supplies. Each type has a linked list and the sub-circuit list is comprised of separate linked lists for the various components (see Appendix B.1).

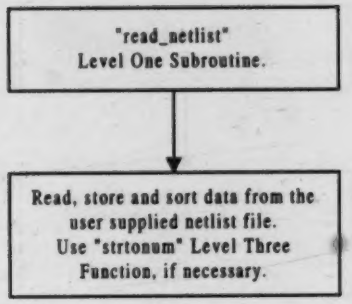

"read_netlist"
Level One Subroutine.
Read, store and sort data from the user supplied netlist file. Use "strtonum" Level Three Function, if necessary.

Figure 4.4: Block diagram of "read_netlist" subroutine.

Numerous tests are used to determine which list a particular information line belongs in. The first test checks for an end of file marker. If it is present the "read_netlist" subroutine ends and the program continues from whence this subroutine was called. If it is not at the end of file, the tests continue.

The next test determines if the first character of data is a period (.). If it is not, the data is assumed to be one of the remaining components, and is entered into the correct list. If the first character is a period, a second test is performed, to check for the form ".subckrt". If it is not, the line of data is purged and the routine continues.

If the data representation is a valid sub-circuit, the data is entered as above if it a normal component and is stored in linked lists under the sub-circuit linked list. This continues until another period is encountered, which defines the end of the sub-circuit. This process is continued until an end of file is encountered.

In some instances, data conversion from ASCII string format to double numeric form may be necessary. This is accomplished by the "strtonum" level-three function. Control returns to the main program, once all of the data is entered.

4.2.3 "cap_calcs" Subroutine

The "cap_calcs" subroutine is called from the main program routine. Its primary purpose, as shown in Figure 4.5, is to call two level-two subroutines, "inter_cap_sort" and "para_cap_calc". These subroutines organize and sort the interconnect capacitances and calculate parasitic capacitances.

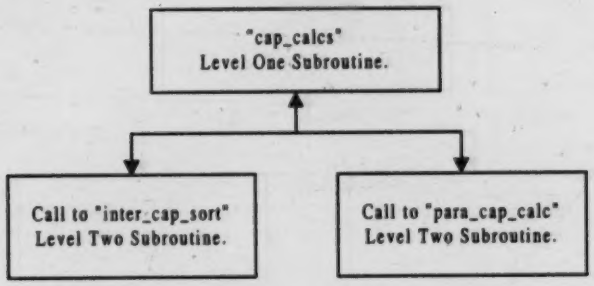


Figure 4.5: Block diagram of "cap_calcs" subroutine.

Control is returned to the main program upon completion of the second subroutine call.

4.2.4 "combine_subccts" Subroutine

The "combine_subccts" subroutine is called from the main program routine. Its purpose, as shown in Figure 4.6, is to call a series of subroutines that create a new series of linked lists. The new linked list structures store the data of the final circuit representation. The structures are defined in the combine.h header file, provided in Appendix B.2

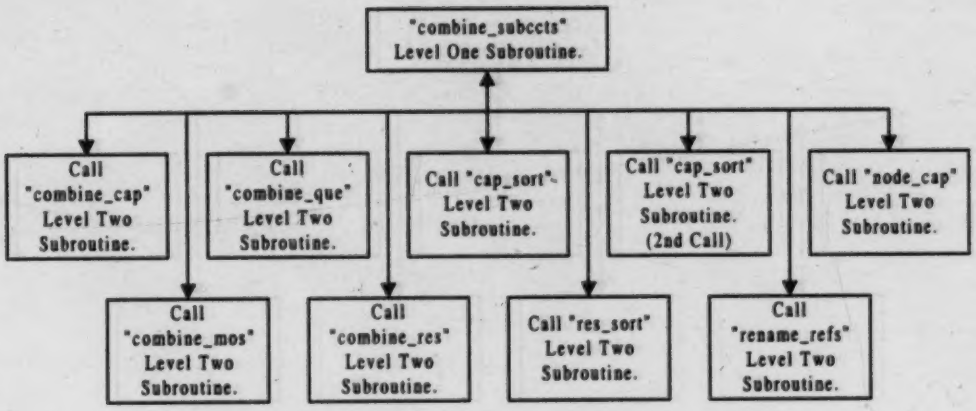

"combine_subccts"
Level One Subroutine.
Call "combine_cap" Level Two Subroutine.
Call "combine_que" Level Two Subroutine.
Call "cap_sort" Level Two Subroutine.
Call "cap_sort" Level Two Subroutine. (2nd Call)
Call "node_cap" Level Two Subroutine.
Call "combine_mos" Level Two Subroutine.
Call "combine_res" Level Two Subroutine.
Call "res_sort" Level Two Subroutine.
Call "rename_refs" Level Two Subroutine.

Figure 4.6: Block diagram of "combine_subccts" subroutine.

Once the new lists are generated, the data is sorted and re-referenced. The end result of executing the subroutines, in the order shown, is to create a single circuit netlist with no sub-circuit references. The next section discusses the purpose and function of these subroutines.

4.2.5 "free_mem" Subroutine

The "free_mem" subroutine is called from the main program routine. The primary purpose of this routine, as shown in Figure 4.7, is to clear and free computer memory that is no longer required.

Figure 4.7: Block diagram of "free_mem" subroutine.

Seven linked list structures, used to store data from the first segment of the program, are no longer required for the analysis and calculations in the remainder of the program. These lists are from the typedefs.h header file, discussed in Appendix B.1. The lists no longer required include: NMOS transistor data, for the NMOS transistor library; PMOS transistor data, for the PMOS transistor library; sub-circuit data, including the sub-circuit capacitor, MOSFET, transistor and resistor data; main circuit capacitor data, main circuit MOSFET data, main circuit transistor data, and main circuit resistor data.

This subroutine uses "free", a standard "C" function call, which dynamically un-allocates the memory previously allocated for the linked lists using a "C" function called "malloc". When all of the unnecessary linked lists have been cleared, the program returns to the main routine.

4.2.6 "time_node_voltage" Subroutine

The "time_node_voltage" subroutine is called from the main subroutine. The purpose of this subroutine, as shown in Figure 4.8, creates, initializes, calculates and stores time and voltage results in data arrays. The arrays are required for the power dissipation calculations in the "power_diss" level-two subroutine.

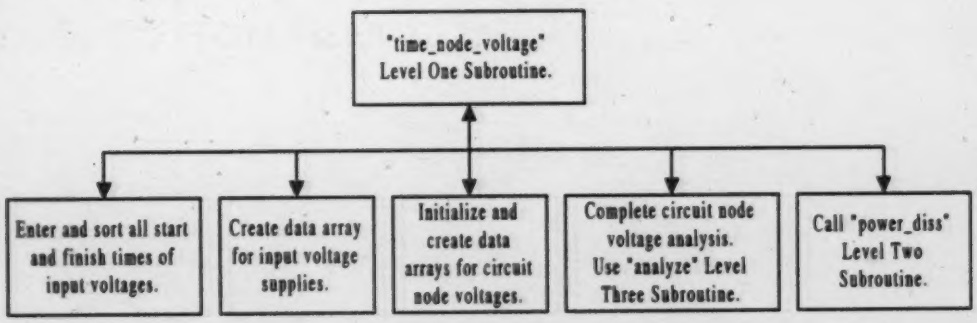

"time_node_voltage"
Level One Subroutine.
Enter and sort all start and finish times of input voltages.
Create data array for input voltage supplies.
Initialize and create data arrays for circuit node voltages.
Complete circuit node voltage analysis. Use "analyze" Level Three Subroutine.
Call "power_diss" Level Two Subroutine.

Figure 4.8: Block diagram of "time_node_voltage" subroutine.

The subroutine can be broken into numerous parts, the first being the creation of a time array. All of the start and finish times of the input voltage supplies are entered and then sorted from smallest to largest.

The next section of the subroutine creates a voltage array for all of the input supplies. This will reflect all of the voltage levels for each time element. An example follows, showing the original netlist voltage entries.

Supply	Start Voltage	End Voltage	Time delay	Pulse Width	Period
V_1	0.0 V	5.0 V	0.5 µs	1.0 µs	2.0 µs
V_2	0.0 V	5.0 V	1.0 µs	1.0 µs	2.0 µs

Table 4.1: Sample Input Voltages from Netlist File

A second chart shows the array resulting from the execution of this subroutine section.

Voltage \ Time	0.0 µs	0.5 µs	1.0 µs	1.5 µs	2.0 µs
V_1	0.0 V	5.0 V	5.0 V	0.0 V	0.0 V
V_2	0.0 V	0.0 V	5.0 V	5.0 V	0.0 V

Table 4.2: Sample Voltage Arrays from the time_node_voltage Subroutine

The third section creates and initializes the overall circuit node (voltage) array. This array stores all voltages for each node at various analysis times. If a particular node is an input, then specify the node voltage for its correct time. The node is reset to a predefined value if it is not an input, as discussed in Section 3.2. In addition, this section also checks for a transistor node (i.e. drain or source) connected to the supply line or ground. The transistor node voltage is set accordingly if this case exists.

The fourth section is the actual voltage analysis of the circuit under test. This section will analyze and flag each node accordingly. The "analyze" level-three subroutine is then called to determine the effects of the transistor node voltages and will set or reset the flags appropriately. Refer to Section 3.2, which discusses this functionality and the analysis subroutine in more detail. The short-circuit currents are also determined at this stage of the analysis.

The final objective of this subroutine is to call the "power_diss" level-two subroutine. This routine determines all power dissipations based on the node voltages calculated and entered in the "time_node_voltage" subroutine. The program returns to the main program, at the successful completion of this routine.

4.3 Level-two Subroutines and Functions

This section discusses all level-two subroutines and functions, which are called from a level-one subroutine. The subroutine flowcharts are provided in Appendix C.3.

4.3.1 "file_check" Function

This function is called from the "read_models" level-one subroutine. If the model library file does not exist or does not contain valid data, an error message is displayed stating the file is invalid and the program terminates. If the file is valid, a value of '1' is returned to the calling point, which is used as a comparison value.

4.3.2 "read_model_data" Subroutine

This subroutine is called from the "read_models" level-one subroutine. All data is entered into linked lists as discussed in typedefs.h header file (see Appendix B.1). In some cases the numerical data is represented in string format, thus requiring the third level function "strtonum" as shown in Figure 4.9.

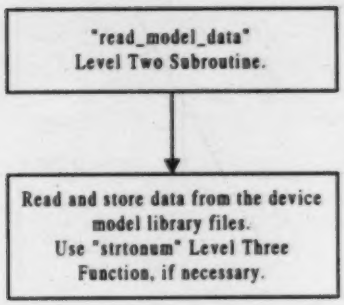

"read_model_data"
Level Two Subroutine.
Read and store data from the device model library files.
Use "strtonum" Level Three Function, if necessary.

Figure 4.9: Block diagram of "read_model_data" subroutine.

"strtonum" converts the ASCII data string into double numerical format. The program is returned to the level-one subroutine from where it was called, when the entries are completed.

4.3.3 "lower_case" Subroutine

The "lower_case" subroutine is called from the "read_models" level-one subroutine. It sets each of the library model linked lists to the head of the list. The level-three subroutine "convert" is then called as shown in Figure 4.10, which converts the model name to lower case ASCII.

Figure 4.10: Block diagram of "lower_case" subroutine.

When the conversion is finished, the linked list pointer increments to its next value. This continues until each library list is scanned then the program returns to where it was called.

4.3.4 "inter_cap_sort" Subroutine

The "inter_cap_sort" subroutine is called from the "cap_calcs" level-one subroutine, and consists of two parts.

The first section scans each sub-circuit. Each interconnect capacitance in a sub-circuit is compared for recurrent parameters. If a match is found, the two capacitances are added and a duplicate or deletion flag is set on the latter entry.

The second section reorganizes the capacitance list for each sub-circuit. The list is scanned for the multiplicity flag, which was set in the first section. If the flag was set, the entry is removed from the list; otherwise, it is left alone. When all lists are checked and sorted, the program is returned to the "cap_calcs" subroutine.

4.3.5 "para_cap_calc" Subroutine

The "para_cap_calc" subroutine called from the "cap_calcs" level-one subroutine has three purposes as outlined in Figure 4.11.

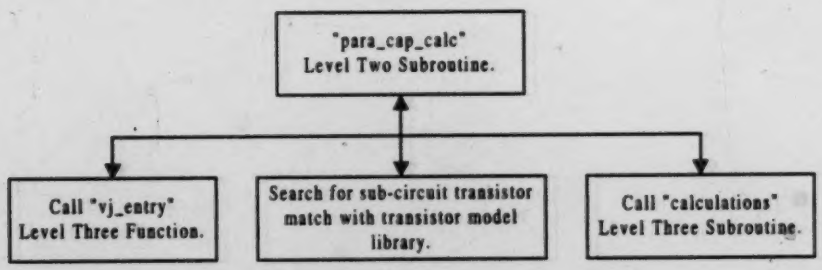


Figure 4.11: Block diagram of "para_cap_calc" subroutine.

The first objective is to call the "vj_entry" level three function, which prompts the user to enter a value for the magnitude of the applied reverse bias junction voltage, V_J. Once a valid number has been entered, the "vj_entry" function returns the entered value to the "para_cap_calc" subroutine, and is then passed to the "calculations" level-three subroutine.

The second purpose is to set the head of the sub-circuit linked list, and scan the model library file for a match with each transistor. For example, if the transistor being tested is an NMOS transistor, using 0.8μ technology, the NMOS library file is selected and scanned for a match in technology name types, i.e. mnch0p8.

The call to the "calculations" subroutine is the final aim of the "para_cap_calc" subroutine. If the transistor type matches the transistor model, then the necessary gate, drain, and source parasitic capacitances can be calculated, as discussed in Chapter 3.1.2.1. If there is no match, an error has occurred. This error is displayed as an error message and the program is terminated.

When the program returnes from the "calculations" subroutine to the "para_cap_calc" subroutine, the next transistor is set and the process repeats. When all transistors in each of the sub-circuits have been tested and the parasitic capacitances calculated, the subroutine is ended and the program is passed back to the calling point in the "cap_calcs" subroutine.

4.3.6 "combine_cap" Subroutine

The "combine_cap" subroutine is called from the "combine_subccts" level-one subroutine. The main purpose of this routine is shown in Figure 4.12(a).

A single complete list of capacitors is generated from each of the sub-circuit lists. Each capacitor node is renamed, with respect to the sub-circuit node list. This new list represents a flat netlist structure without sub-circuit data. When the new capacitor list is completed, the program returns to its calling place in the "combine_subccts" subroutine.

4.3.7 "combine_mos" Subroutine

The "combine_mos" subroutine is called from the "combine_subccts" level-one subroutine. The primary purpose of this routine is shown in Figure 4.12(b).

A single complete list of MOSFETs is generated from the sub-circuit lists. The drain, gate, source, and substrate nodes of each MOSFET is renamed, with respect to the sub-circuit node list. This new list, as with the "combine_caps" subroutine, represents a flat netlist structure without sub-circuit data. When the new MOSFET list is completed, the program returns to the "combine_subccts" subroutine.

4.3.8 "combine_que" Subroutine

The "combine_que" subroutine is called from the "combine_subccts" level-one subroutine. The primary purpose of this routine is shown in Figure 4.12(c).

A single complete list of transistors is generated from the sub-circuit lists. The collector, base, emitter, and substrate nodes of each transistor is renamed, with respect to the sub-circuit node list. This new list also represents a flat netlist structure without sub-circuit data. When the new transistor list is completed, the program returns to the "combine_subccts" subroutine.

4.3.9 "combine_res" Subroutine

The "combine_res" subroutine is called from the "combine_subccts" level-one subroutine. The primary purpose of this routine is shown in Figure 4.12(d).

A single complete list of resistors is generated from the sub-circuit lists. Each resistor node is renamed, with respect to the sub-circuit node list. This new list, as in the three previous routines, represents a flat netlist structure without sub-circuit data. When the new resistor list is completed, the program returns to the "combine_subccts" subroutine.

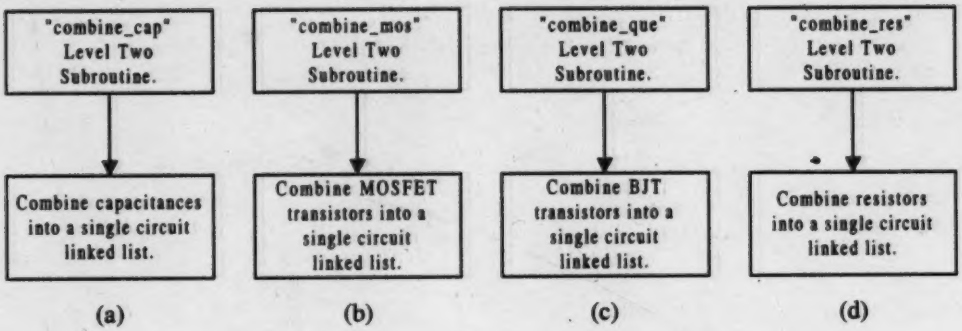

"combine_cap" Level Two Subroutine.
"combine_mos" Level Two Subroutine.
"combine_que" Level Two Subroutine.
"combine_res" Level Two Subroutine.
Combine capacitances into a single circuit linked list.
Combine MOSFET transistors into a single circuit linked list.
Combine BJT transistors into a single circuit linked list.
Combine resistors into a single circuit linked list.
(a)
(b)
(c)
(d)

Figure 4.12: Block diagrams of (a) "combine_cap" (b) "combine_mos" (c) "combine_que" (d) "combine_res" subroutines.

4.3.10 "cap_sort" Subroutine

The "cap_sort" subroutine is called from the "combine_subccts" level-one subroutine. This subroutine combines and sorts all interconnect capacitances of a single capacitor linked list.

Two loops are required to complete this stage of the program. The first scans the capacitor linked list for multiple instances of capacitors. If a multiple instance occurs, its value is added to the first instance and a duplicate flag is set. If one of the capacitor nodes connects to V_{DD} or ground, its duplicate value is added to the first instance of the node and a duplicate flag is set. If one node is connected to V_{DD}, it is changed to a ground reference; otherwise, it remains unchanged.

The second loop scans the linked list for the capacitors with the duplicate flag set. If a flag is set, the linked list entry is eliminated. This process continues until the entire list is examined. The program returns to the "combine_subccts" subroutine.

4.3.11 "res_sort" Subroutine

The "res_sort" subroutine is called from the "combine_subccts" level-one subroutine.

This routine is almost identical to the "cap_sort" level-two subroutine with a couple of differences. The first: instead of the capacitor list, the resistor list is scanned for multiple instances.

The next difference is with the first loop. Only one test is necessary to determine multiple entries. A duplicate flag is set only if the nodes of both the first resistor and the current resistor match. The second loop does not differ from the "cap_sort" routine. The list is scanned for the setting of the duplicate flag. If the flag is set, the linked list entry is removed. At the completion of the scan, the program is returned to the "combine_subccts" subroutine.

4.3.12 "rename_refs" Subroutine

The "rename_refs" subroutine is called from the "combine_subccts" level-one subroutine. The main purpose of this routine is to scan through each of the newly created data lists and rename all device reference names. This subroutine prevents multiple instances of component reference names from occurring.

The sub-circuit data contains instances of all components, reference names, and values. Multiple components with the same reference will exist if any sub-circuit data is used multiple times. This subroutine avoids conflicts in reference names and creates a new naming scheme, referring to the old sub-circuit, for the new circuit netlist. When this subroutine is completed, the program returns to the "combine_subccts" subroutine.

4.3.13 "node_cap" Subroutine

The "node_cap" subroutine is called from the "combine_subccts" level-one subroutine. Only the MOSFET transistor list is required for analysis of the nodes, since all of the transistors are directly connected to each other, in pure CMOS circuits. This subroutine is used to total all transistor node capacitances and individual node capacitances of the final circuit netlist and store the data in a new node-only linked list. There are four goals of this subroutine.

The first loop checks each drain, gate, and source for the node and capacitance associated to it, so long as the node is not V_{DD}, or V_{SS} (ground for most digital CMOS circuits). All of the results are stored in a newly created linked list.

The second loop scans the generated linked list and combines the total MOSFET capacitance for each node and leaves the result in a single node reference. Like the "cap_sort" and "res_sort" subroutines, duplicate node entries are flagged and the total capacitance is stored in the first instance of a node.

The third loop removes the multiple nodes listed and flagged in the second loop. When this is completed, the list is set to be combined with equivalent interconnect capacitances from the capacitance list, which is the final loop.

For each final entry of the node linked list, the capacitance list is scanned for a matching interconnect capacitance. The matching interconnect capacitance must be connected between the node under test and ground (or V_{SS}). If this condition is met, the interconnect capacitance is summed with the existing node capacitance. A flag is also set on the capacitor in the capacitance list.

When the "cap_sort" is called a second time from the "combine_subccts" subroutine later in the program, the duplicate entry will then be removed. After the entire node list is scanned, the program returns to the "combine_subccts" subroutine.

4.3.14 "analyze" Subroutine

The "analyze" subroutine is called from the "time_node_voltage" level-one subroutine. The primary goal of this subroutine is to evaluate the drain, source, and gate of each MOSFET transistor in the circuit.

When this subroutine is called, a single node of a transistor is under analysis. Changes in this node may affect the operation of the transistor nodes in the reminder of the circuit. As the various gate, drain and source nodes of each remaining transistor are analyzed, the necessary voltages are set. The flag of each affected node is set as to the

current state of the MOSFET. This flagging approach was discussed in Section 3.2. When the analysis of the circuit is completed for the node under test, the program is returned to the "time_node_voltage" subroutine.

4.3.15 "power_diss" Subroutine

The "power_diss" subroutine is called from the "time_node_voltage" level-one subroutine and consists of three parts, as outlined in Figure 4.13.

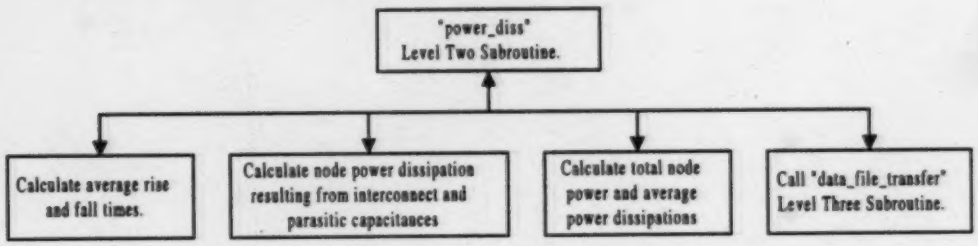

"power_diss"
Level Two Subroutine.
Calculate average rise and fall times.
Calculate node power dissipation resulting from interconnect and parasitic capacitances
Calculate total node power and average power dissipations
Call "data_file_transfer" Level Three Subroutine.

Figure 4.13: Block diagram of "power_diss" subroutine.

The first part of this subroutine can be broken into five sections. The first calculates the average rise and fall times from all of the supplies listed in the netlist. These average times are required for the computation of the various power dissipations. The second section determines the dynamic energy consumption and power dissipation of each node due to parasitic capacitances. The third section calculates the dynamic energy consumption and power dissipation of each node due to interconnect capacitances. The fourth section computes the total dynamic power dissipation of each node (the sum of the interconnect and parasitic powers) and the total short-circuit power dissipation. The fifth section calculates the average power dissipation of each node for the entire test period, by summing the dynamic and short-circuit power contributions.

The second main portion of this subroutine outputs all calculated data to a file. The data is in Matlab matrix format, including: node labels, time labels, time, node

parasitic power, node interconnect power, total node power, average node power, node voltage, and maximum time and node number.

The final part of this subroutine calls the "data_file_transfer" level-three subroutine. This appends the Matlab executable code to the output data file. When the stages are finished, the program returns to the "time_node_voltage" subroutine.

4.4 Level-three Subroutines and Functions

This section discusses all level-three subroutines and functions, which are called from level-two subroutines and/or function routines. The flowcharts for level three subroutines and functions are provided in appendix C.4.

4.4.1 "strtonum" Function

The "strtonum" function is called from two places, the "read_model_data" level-two subroutine, and the "read_netlist" level-one subroutine. This function removes all extraneous string data up to and including the equal sign (=). The remaining string, representing a numerical value, is converted to double format using the "C" preprocessor function "strtod" and returns the value to its calling place for storage in its appropriate linked list.

4.4.2 "convert" Subroutine

This subroutine is called from the "lower_case" level-two subroutine. It changes each character of the data string scanned in from the "lower_case" subroutine. The conversion from uppercase to lowercase ASCII is completed using the "C" preprocessor function "tolower". At the completion of the conversion, program control is returned to the location from which this routine was called.

4.4.3 "vj_entry" Function

The "vj_entry" function is called from the "para_cap_calc" level-two subroutine. This function prompts the user to enter a value for V_J, which is the magnitude of the applied reverse bias junction voltage. Once this value has been successfully entered, it is returned to the "para_cap_calc" subroutine.

"vj_entry" requires the user to enter a voltage magnitude between 0.000V and the supply, followed by the sign of the magnitude, positive (+) or negative (-). The subroutine checks the validity of the number entered, which must be correct prior to the user prompt for the sign of the magnitude. If the number is invalid, i.e. an alphanumeric is present, the subroutine prompts the user to enter another value and will continue until a legitimate value is entered.

If the sign is incorrect, the user will be re-prompted to enter the sign until it is legitimate. Once the number and sign are correct, the program returns the value to its calling point in the "para_cap_calc" subroutine.

4.4.4 "calculations" Subroutine

The "calculations" subroutine is called from the level-two, subroutine "para_cap_calc". Its primary purpose is shown in Figure 4.14 and the flowchart is referenced in Appendix C.4.

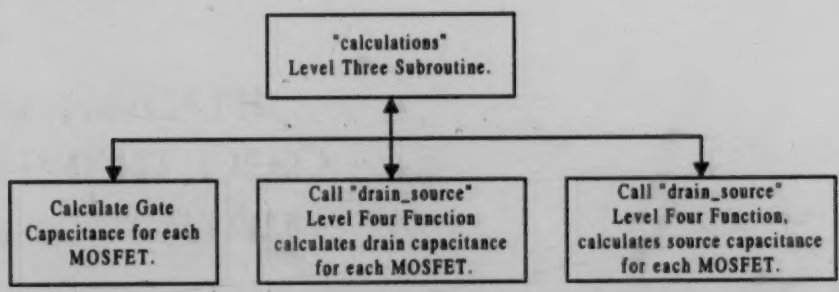

"calculations"
Level Three Subroutine.
Calculate Gate Capacitance for each MOSFET.
Call "drain_source" Level Four Function calculates drain capacitance for each MOSFET.
Call "drain_source" Level Four Function, calculates source capacitance for each MOSFET.

Figure 4.14: Block diagram of "calculations" subroutine.

This subroutine calculates the gate, drain, and source capacitances for each transistor in a sub-circuit list. The drain and source capacitances use the same formula for their respective values, thus the level-four "drain_source" function was created to do the calculations. Once the required parameters are computed and stored, the program returns to its calling point in the "para_cap_calc" subroutine.

4.4.5 "data_file_transfer" Subroutine

This subroutine is called from the "power_diss" level-two subroutine. Its primary purpose is to copy a data file into the output file, as specified by the user in the main "power" program routine.

. If the data file to be transferred does not exist in the present working directory, an error message is displayed and the program is terminated. If the data file is corrupted or does not contain any data, an error message is displayed and the program is terminated.

If the data file is valid, the file is opened and the data is transferred to the user-supplied output file. When all data is exported, the data file is closed and the program returns to the "power_diss" subroutine.

4.5 "drain_source" Level-four Function

The "drain_source" is the only level-four function which is called from the "calculations" level-three subroutine. This function calculates the drain or source capacitance for a transistor. The formula for these capacitances is the same, therefore only the specific area and perimeter data, device parameter data and junction voltage need to be passed, via the function call.

The drain or source capacitance consists of two parts. The first portion is the area contribution and the second is the perimeter contribution, which are described by Equations 3.7 and 3.8. The total capacitance is calculated using Equation 3.6. When all

calculations have been performed, the resulting total capacitance is passed back to the "calculations" subroutine.

4.6 Matlab Program Data File

The discussion of the "data_file_transfer" level-three subroutine in Section 4.4.5 makes reference to copying a data file to the output results file. This data file is the Matlab source code that is the main executable program, when run in the Matlab simulation program.

The source-code of the Matlab program is provided in Appendix C.5. Figure 4.15 shows the major functional components of the source code.

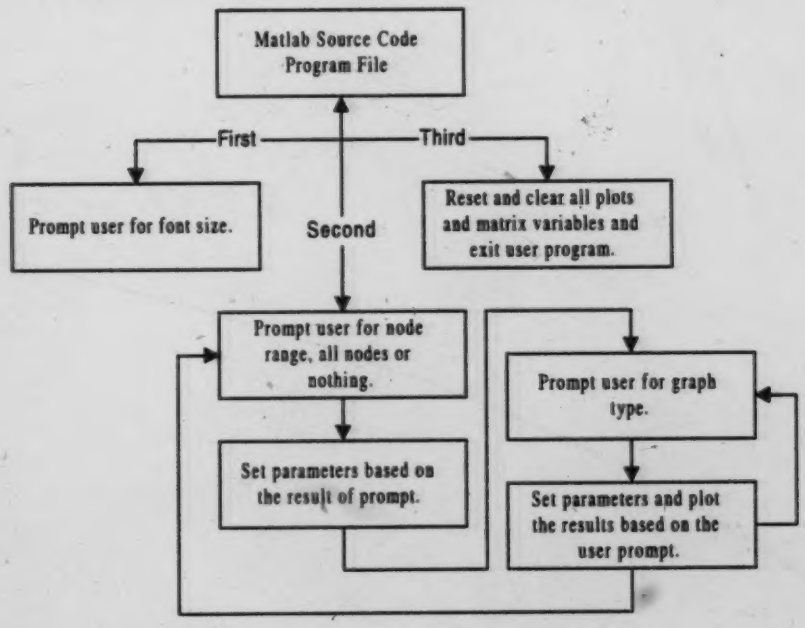

Matlab Source Code Program File
First
Second
Third
Prompt user for font size.
Reset and clear all plots and matrix variables and exit user program.
Prompt user for node range, all nodes or nothing.
Set parameters based on the result of prompt.
Prompt user for graph type.
Set parameters and plot the results based on the user prompt.

Figure 4.15: Block diagram of the Matlab source code operation.

The first Matlab program function prompts the user for the desired font size of the graphs. The prompt remains until a valid answer between 6 pt. and 12 pt. is entered.

Once a font size is entered, any unnecessary variables are cleared from the program memory. Figure 4.16 below shows the font menu structure during program execution.

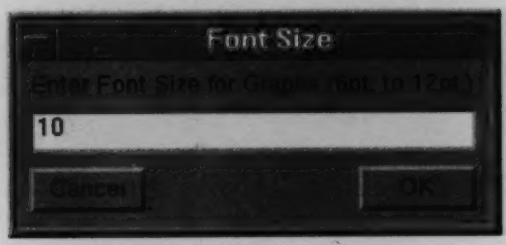


Figure 4.16: Font size menu GUI.

The second function prompts the user to select which node(s) of the circuit are of interest to plot. There are three choices, as shown in Figure 4.17. They include a range, all, or none of the nodes.

Figure 4.17: Menu of nodes to plot GUI.

If a range of nodes is selected from the menu, the user must enter a start and end node. Sample menus are shown in Figure 4.18, in which the start menu (a) and end menu (b) nodes are push-button selectable.

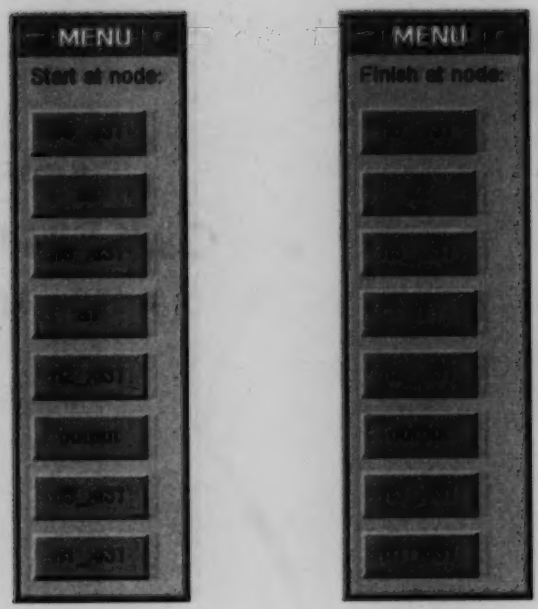


Figure 4.18: (a) Start node menu GUI, (b) End node menu GUI.

When the start and end nodes, or all nodes have been selected, the data matrices from the ***power*** program are copied into new temporary arrays for manipulation. The program continues to the third function once all of the parameters are set.

If the 'end' button of Figure 4.17 was selected, the user is prompted to confirm the exit of the program. The menu selection is shown in Figure 4.19.

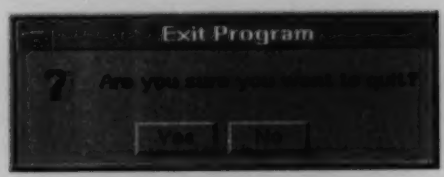


Figure 4.19: Exit menu GUI.

If 'no' is selected, the program resets the second function. If the response is 'yes', then it resets and cleans any existing plots, and all matrix variables used by the Matlab user program. When the memory is cleared, the user program ends.

The third Matlab program function prompts the user to select a plot to view. The user can select the plot representations of the data as shown in Figure 4.20. The plot types include the node voltages, the node power dissipation due to parasitic capacitances, the node power dissipations due to interconnect capacitances, the total power dissipated for all time elements, the total average power dissipated by the circuit; and finally, the user can choose to exit.

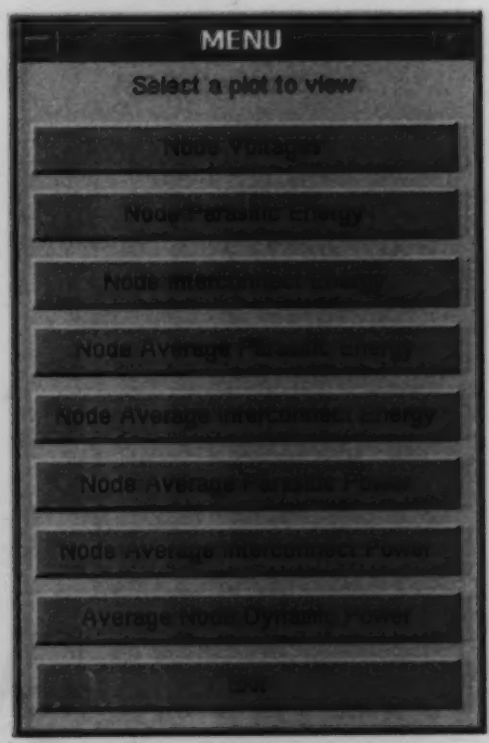


Figure 4.20: Plot type menu GUI.

If 'exit' is not selected, the user can choose to display any of the five plots concurrently, in various figure windows.

When a graph is displayed, the user has the option of rotating the resulting graph for easier viewing. When the analysis of a graph is completed, the user can close the figure window without affecting any other windows that may be open.

A complete example of the program operation, including the Matlab results is described in Chapter 5.

5.0 Results

This chapter discusses the model test circuits used to demonstrate a sample application session, presents results for total power and dynamic power dissipation, and addresses the program efficiency.

5.1 Test Circuits

Sixteen test circuits were sampled to examine the effectiveness of the ***power*** program. Twelve of these circuits consist of a single two-input logic gate. The logic gates include the standard AND, NAND, OR, NOR, XOR, and XNOR configurations. The remaining four circuits are more complex designs including a 1-bit adder and a 4-2 compressor circuit. The netlist files for the XNOR (T-Cell) gate and the 4-2 compressor (K-Cell) circuit are found in Appendices D1 and D3 respectively.

The netlist of each sample circuit was created using two different cell models. The K-Cell and T-Cell Cadence models differ in many ways. The K-Cell is smaller and faster than the T-Cell, the number of transistors is fewer and the width to length ratio is smaller. K-Cells are meant to operate at a supply voltage of 3.3V and 5.0V and the T-Cell is designed to operate strictly at 5.0V, as specified by the technology and cell layout. The changing effects of power consumption are demonstrated using the different circuit configurations and supply voltages suggested.

5.2 Sample Application Session

The data for this sample session were generated from a two-input T-Cell XNOR gate, which was chosen since it is the most complex of the standard cells. A schematic representation of the circuit is shown in Appendix D2.

The first step is to execute the ***power*** program by entering the following command line at the UNIX prompt:

power <netlist> <output.m>

After the program has terminated without errors, the *output.m* file can be executed using the Matlab program. Figures 5.1 to 5.8 show the dynamic energy and power output results generated graphically by Matlab for the sample circuit.

Figure 5.1 shows the voltage for all nodes at each transition of time. The z-axis represents the voltage (Volts); the y-axis denotes each time interval in which a transition of the input has occurred (seconds); and the x-axis signifies the node label. In this example, the supply and input voltages are set at five volts. A node voltage between zero and five volts results from the threshold voltage and the non-standard configuration of the NMOS or PMOS transistor as discussed in Chapter 3.

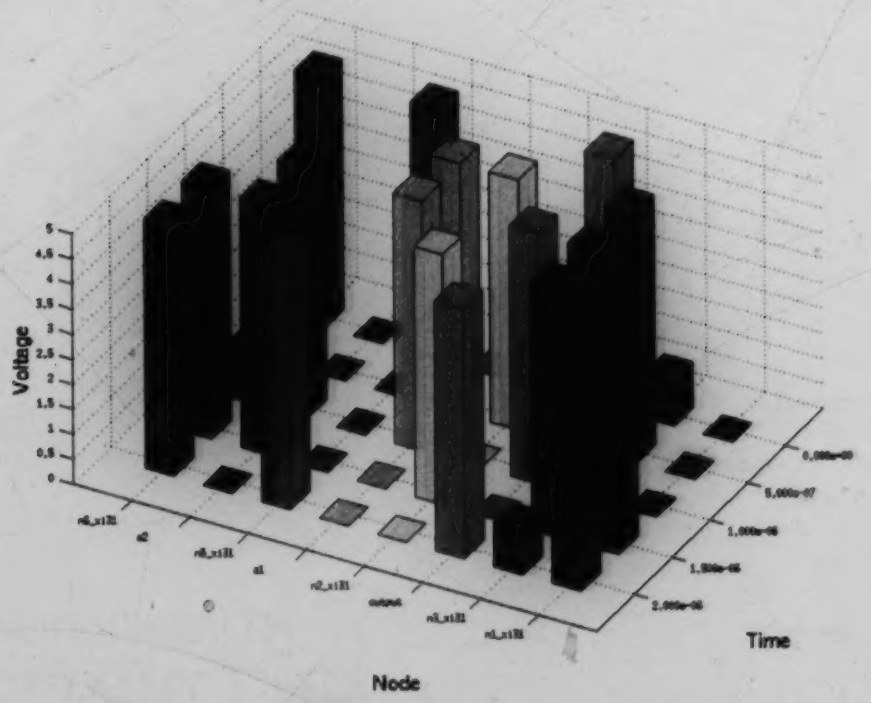


Figure 5.1: Voltage transitions for each node and time transition.

Figure 5.2 shows the amount of energy consumed by each node due to the effects of parasitic capacitances. Figure 5.3 shows the contribution of energy consumed by each node due to the interconnect capacitances in the layout. The z-axis represents the energy consumed by each node (Joules); the y-axis represents each time interval in which a transition of the input has occurred (seconds); and the x-axis denotes the node labels.

One factor resulting in a lower energy contribution to the node, is that the parasitic capacitance is generally much larger than the interconnect capacitance. Another factor for a lower energy contribution, due to interconnect capacitances, results from differences between two nodes with different voltage potentials. This potential difference is not between a node and ground, as is the case with the parasitic capacitance.

This data are of extreme importance. The amount of energy consumed over a specific time interval denotes the power dissipated by the node.

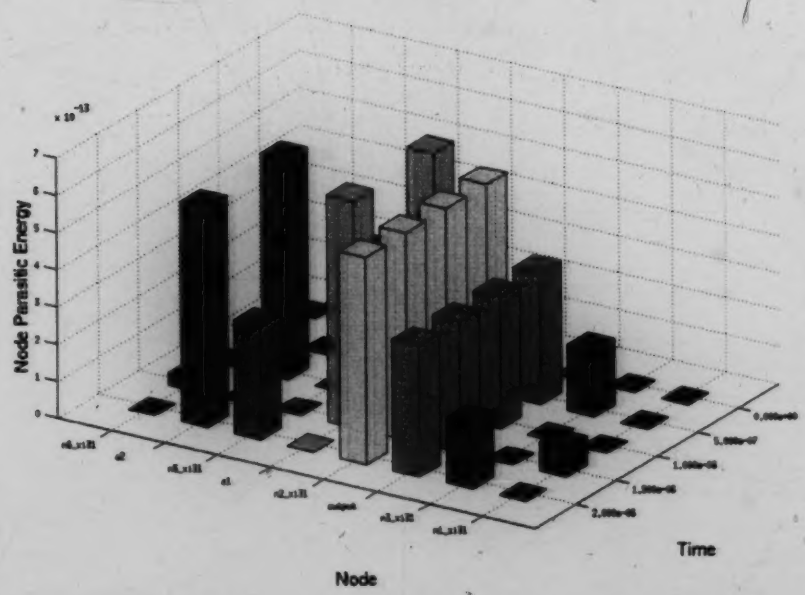


Figure 5.2: Parasitic energy consumed at each node for each time transition.

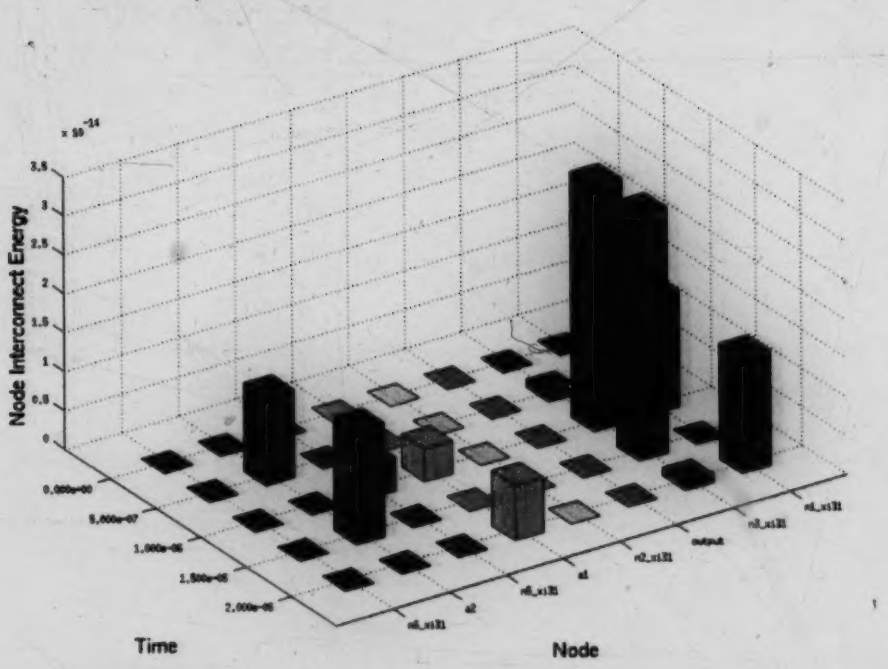


Figure 5.3: Interconnect energy consumed at each node for each time transition.

Figures 5.4 and 5.5 present the results of the average energy consumed by each node of the circuit during the total test period. The energy contribution of each node is summed and divided by the number of event transitions, as discussed in Chapter 3.

Figure 5.4 shows the energy consumption from the contribution of parasitic capacitances at the node under examination. Figure 5.5 shows the energy consumed from the contribution of interconnect capacitances at the various nodes under test. The z-axis represents the total average energy consumed by each node (Joules); the y-axis represents the total time interval in which the test vectors occur (seconds); and the x-axis denotes the node labels.

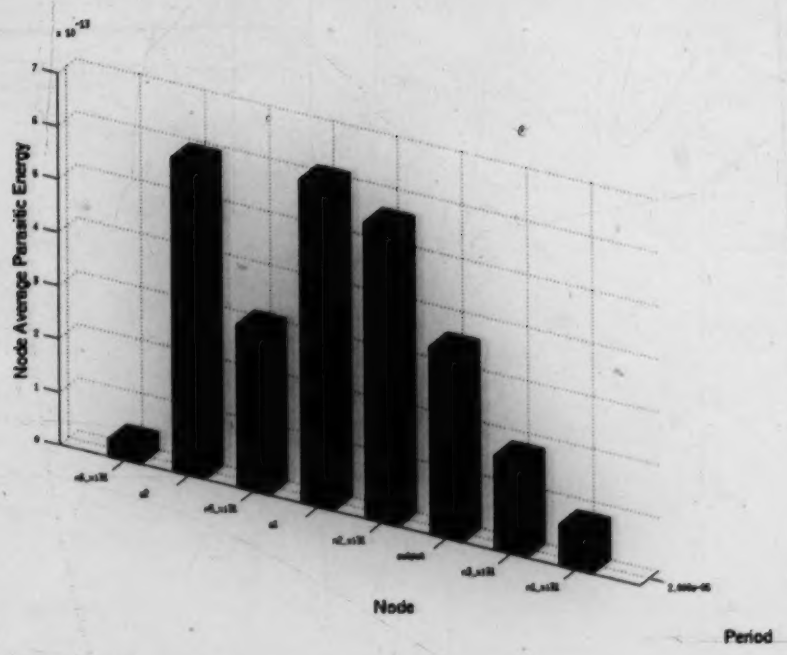


Figure 5.4: Average energy consumed at each node due to the parasitic capacitances.

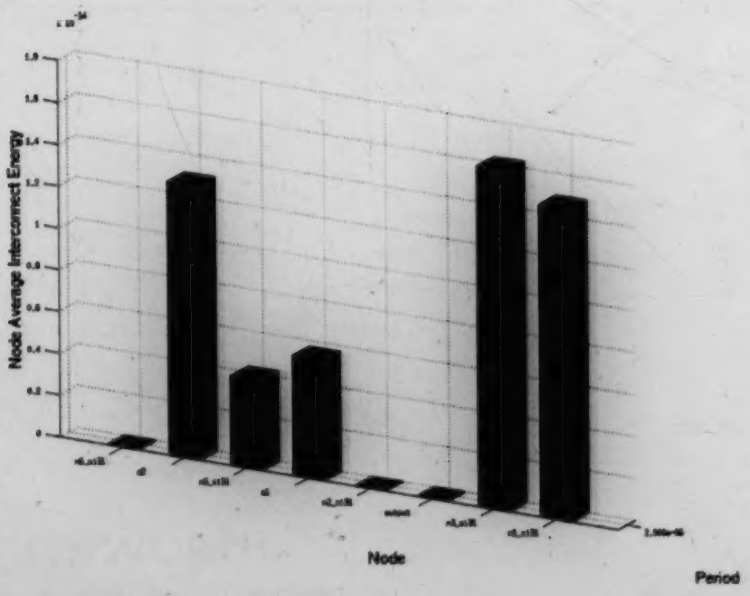


Figure 5.5: Average energy consumed at each node due to the interconnect capacitances.

Figures 5.6 and 5.7 represent the average dynamic power dissipated by each node of the circuit during the total test period. The results are directly proportional to the energy consumed at each node. The power is calculated by dividing the energy consumed by the period of test.

Figure 5.6 shows the average dynamic power dissipated at each node due to the parasitic capacitance contribution. Figure 5.7 presents the average dynamic power dissipated at each node due to the contribution of interconnect capacitances. The z-axis signifies the total average dynamic power consumed by each node (Watts); the y-axis denotes the total time interval in which the test vectors occur (seconds); and the x-axis represents the node labels.

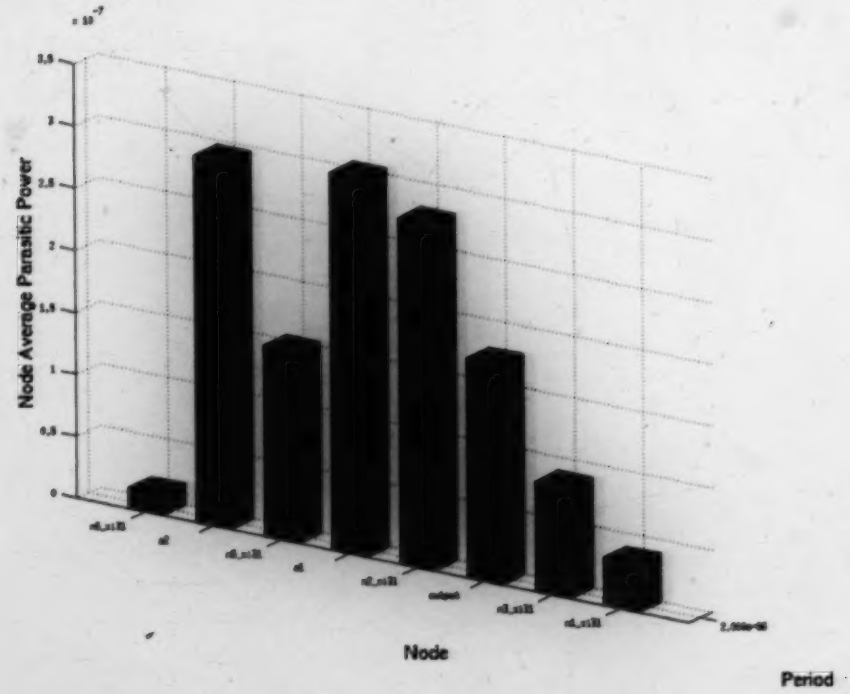


Figure 5.6: Average dynamic power (parasitic contribution) consumed at each node.

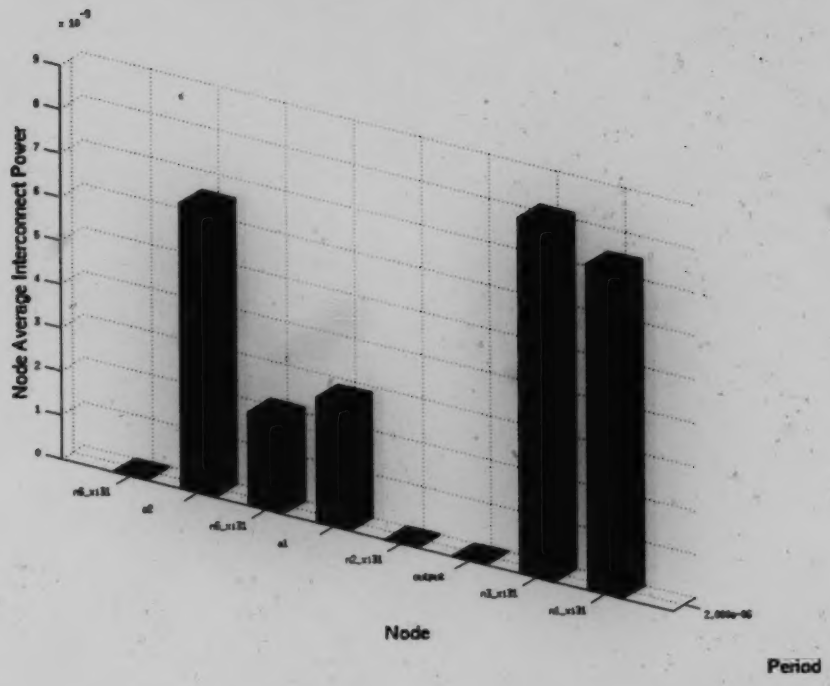


Figure 5.7: Average dynamic power (interconnect contribution) consumed at each node.

Figure 5.8 shows the total average dynamic power dissipated by each node. The parasitic and interconnect capacitance contributions cause this dynamic power. It is the direct result from switching transitions at each node of the circuit, during the total test period. The z-axis represents the total average dynamic power consumed by each node (Watts); the y-axis is the total time interval in which the test vectors occur (seconds); and the x-axis signifies the node labels.

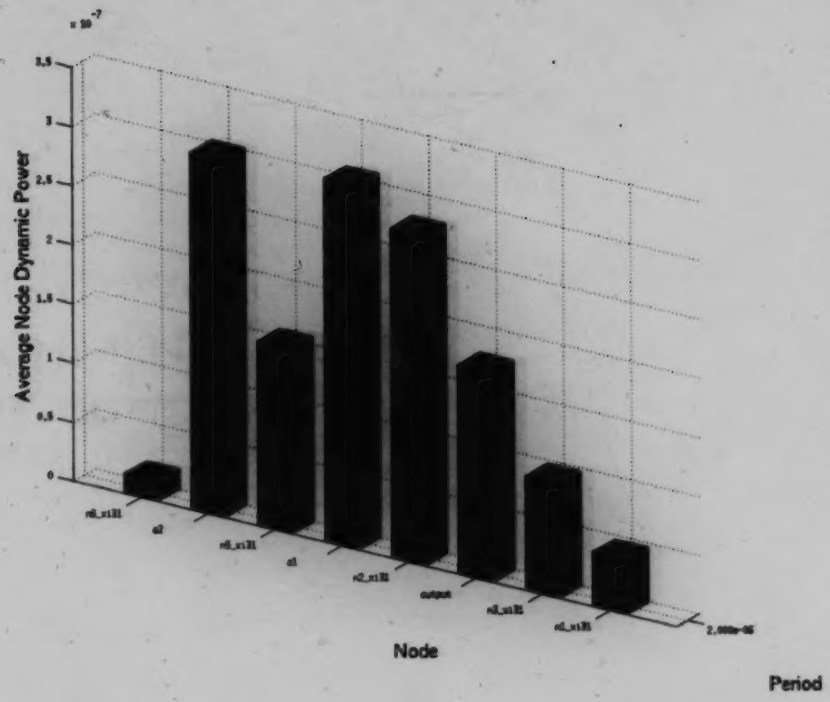


Figure 5.8: Total average dynamic power consumed at each node.

Additional results are calculated but not graphically represented. First, the total average dynamic power dissipated is derived from the summation of all node powers, as shown in Figure 5.8, and divided by the total number of nodes. Second, the short-circuit contribution to the total power is calculated from Equation 3.19. Finally, the total average power dissipated by the device is calculated by summing the total average dynamic power and the total short-circuit power dissipations. These results are discussed in further detail in the next section.

5.3 Data Results

This section examines and compares the results of the total power dissipations from various devices, using Analog Artist software and the ***power*** program. The ***power*** program results are also reviewed to determine if the short-circuit power contribution confirms theoretical results.

5.3.1 Total Power Results

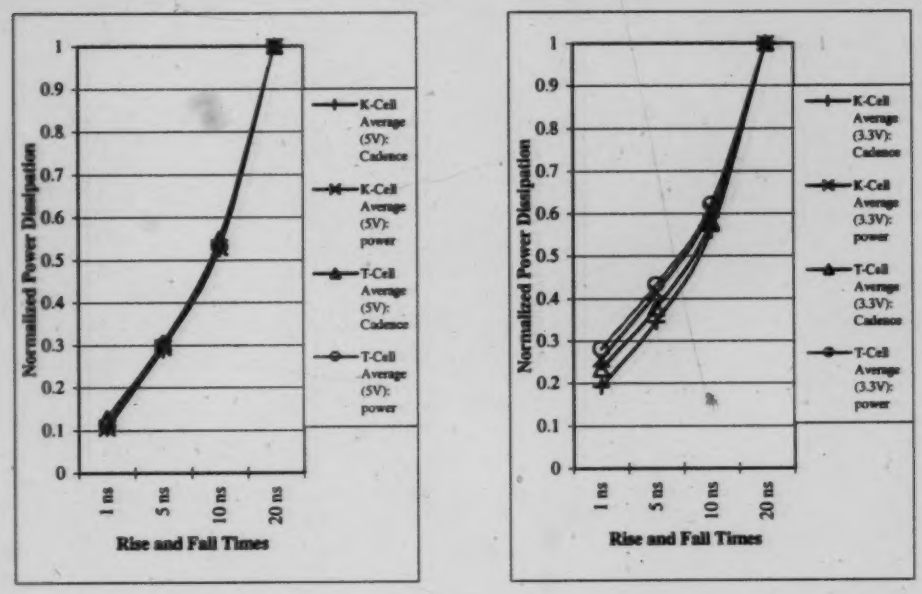
Parallel simulations were conducted using the Analog Artist tool and the ***power*** program, to verify the correct operation of the software. The Analog Artist simulations describe the effects of the average current of the voltage supply. Thus, the total average dissipated power is comprised of the short-circuit and dynamic dissipations, as well as the leakage currents. The ***power*** program consists of the short-circuit and dynamic dissipations only.

The tests were completed using 3.3V and 5V supply voltages respectively. Four rise and fall times (1ns, 5ns, 10ns and 20ns) were also tested at each voltage level. Figures 5.9 and 5.10 show the normalized results between the Analog Artist simulations and the ***power*** program.

The simulations show trends that are consistent with theoretical results [26]. The total power dissipation increases exponentially at first, then linearly as the rise and fall times lengthen. The dynamic power is independent to changes in rise and fall times as discussed in Chapter 3, and thus remains constant. The short-circuit power contribution depends on the rise and fall times of switching transitions; thus the total power increases as expected.

The individual gates have similar power dissipation patterns. Therefore, Figure 5.9 shows the results for the average of all single gate circuits. Figure 5.9(a) shows the normalized power dissipation by the T-Cell and K-Cell gate configurations at input and

supply voltages of 5V. Figure 5.9(b) shows the normalized power dissipation by the T-Cell and K-Cell gate configurations at input and supply voltages of 3.3V.



(a) (b)

Figure 5.9: Normalized power dissipation curves for standard T-Cell and K-Cell gates with: (a) 5V supply voltage and (b) 3.3V supply voltage.

The Cadence and ***power*** program results for a single gate T-Cell and K-Cell circuit are almost identical, when compared for a supply voltage of 5V. The results are slightly different for a supply voltage of 3.3V, however the pattern remains constant. The T-Cell structures were designed to work with a supply voltage of 5V, whereas the K-Cell gates were designed to work with a supply voltage between 3.3V and 5V. This difference explains the increase in power dissipations at 1ns, for a 3.3V supply, compared to that of the 5V supply.

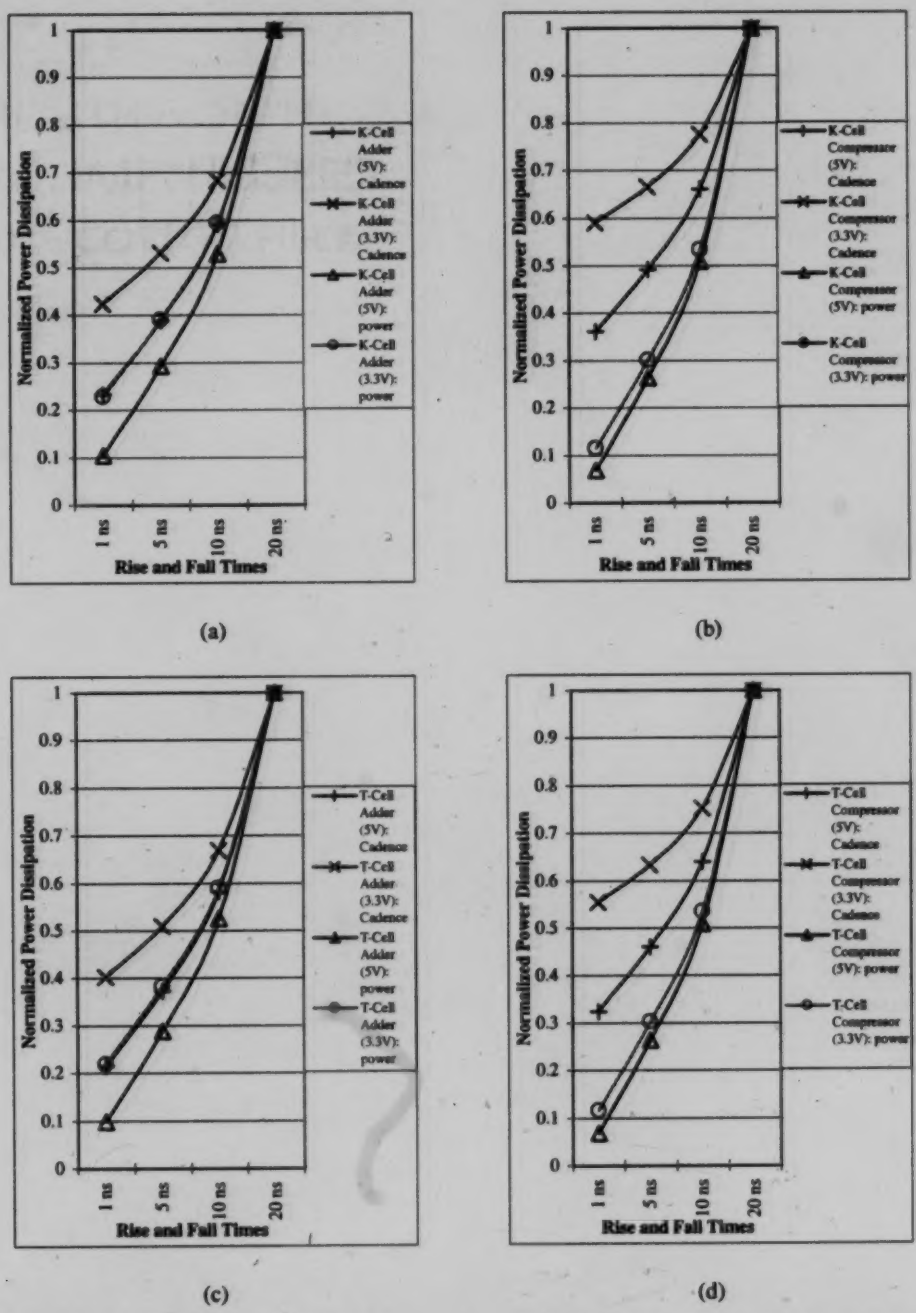


Figure 5.10: Normalized power dissipation curves using 3.3V and 5V supply voltages for: (a) K-Cell adder circuit, (b) K-Cell compressor circuit, (c) T-Cell adder circuit, and (d) T-Cell compressor circuit.

Figure 5.10 shows the normalized power results for a 1-bit adder and a 4-2 compressor circuit in T-Cell and K-Cell configurations. The circuits are more complex than the single gate circuits, resulting in high power dissipations. The normalized results for the T-Cell and the K-Cell configurations of each circuit are comparable i.e. Figure 5.10(a),(c) and Figure 5.10(b),(d). As well, the overall pattern still follows the theoretical results that the power dissipation increases as the rise and fall times increase.

There is a more noticeable difference between the Analog Artist and ***power*** results for the complex circuits when compared to the single gate circuits. The discrepancies between Analog Artist and ***power*** can be attributed to a number of factors. The most important difference is that ***power*** uses level one and two Spice approximations, whereas Cadence uses complete Spice statistical modeling. This is especially noticeable when calculating the short-circuit and static power dissipations.

Equal rise and fall times of the input and output signals are assumed by the ***power*** program at 1ns. The short-circuit power contribution will be less than 20% of the total power if this assumption is true. Discrepancies between Analog Artist and ***power*** can result if Analog Artist has the output signal rise and fall times shorter than the input rise and fall times. If this happens, there can be a significant increase in the short-circuit contribution; i.e., the short-circuit power can increase to the dynamic power [26].

Charge sharing effects of the capacitances is another factor not accounted for in the ***power*** program. Delay factors for decreasing voltage supplies were also neglected. When supply voltages drop to near threshold voltage levels, the transistor switching time decreases, independent of the rise and fall times of the input. This may cause faulty operation of the transistor and thus an increase in the power dissipation. Analog Artist accounts for this phenomenon, the ***power*** program does not.

5.3.2 Dynamic Power Results

The results presented in Figure 5.11 denote the total dynamic power as a percentage of the total power dissipated by the device under test. The final results for

the standard gates show similar trends for both the K-Cell and T-Cell variations using a supply voltage of 5V. Thus, an average was taken for all K-cell gates and T-Cell gates. A common pattern exists between the standard gates and complex circuits tested using ***power***. The exponential decay in the total percentage of dynamic power, Figure 5.11, is consistent with Equation 3.19.

The graph shows the short-circuit contribution to be less than 20% of the total power dissipation for significantly shorter input rise and fall times [26]. As the rise and fall times increase, the short-circuit power contribution becomes more pronounced.

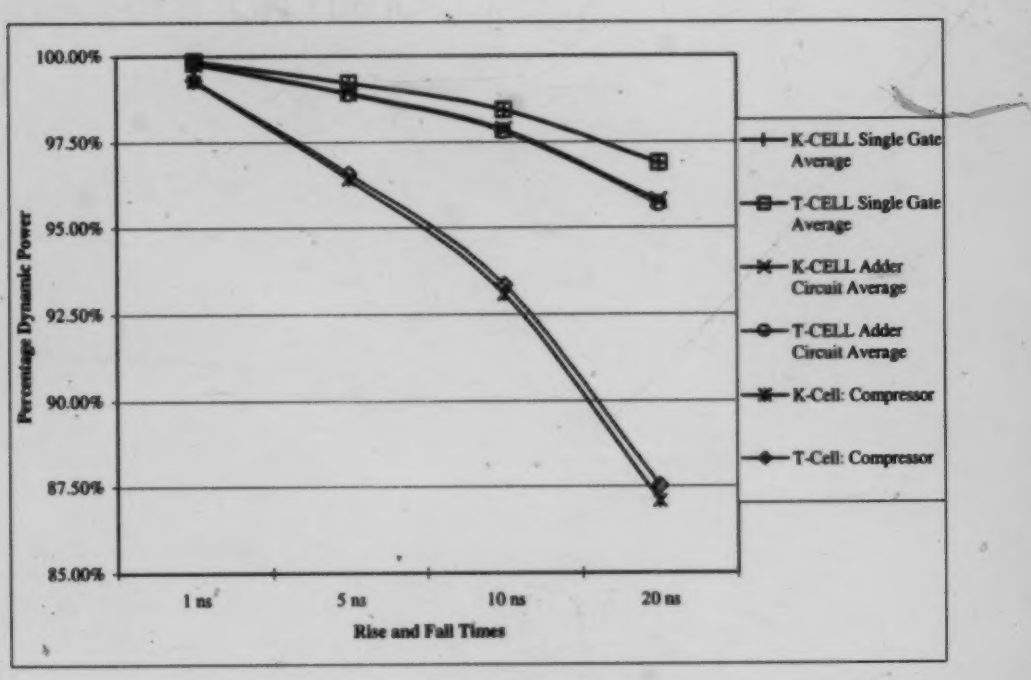


Figure 5.11: Contribution of Dynamic Power to Total Power Dissipation based on the power program for the test gates and circuits.

5.4 Program Efficiency

The cost incurred by the execution of a computer program, or algorithm is determined by the time required to compute the result(s) and the amount of storage space the program needs to effectively arrive at the result(s). The time requirement is the most important cost for this software application, as the space requirements will change depending on the user data. The time complexity of an algorithm or entire program, for example *f(n)*, is said to be of order *f(n)* and is denoted O(*f(n)*).

The structure of the ***power*** program does not provide a means to effectively estimate an overall efficiency or order. There is also no comparison available for this program since the data varies depending on the user supplied netlist file. For these reasons the efficiency is considered for the node voltage analysis and power dissipation analysis stages of the program. The worst case scenario is also used when discussing the time complexity.

The voltage analysis stage tends toward $O(m^2)$, where m is the total number of transistors in the final linked list. Other efficiencies are present; however, they tend to be linear in nature. Although $O(m^2)$ is not a desired algorithm for large values of m, it is a realistic efficiency for early revisions of this software. This ensures correct program operation.

The power dissipation analysis stage tends toward O(n) where n is the number of nodes in the circuit. Other efficiencies are present, and each is linear. Therefore, the worst case efficiency is O(n) for this stage of the analysis.

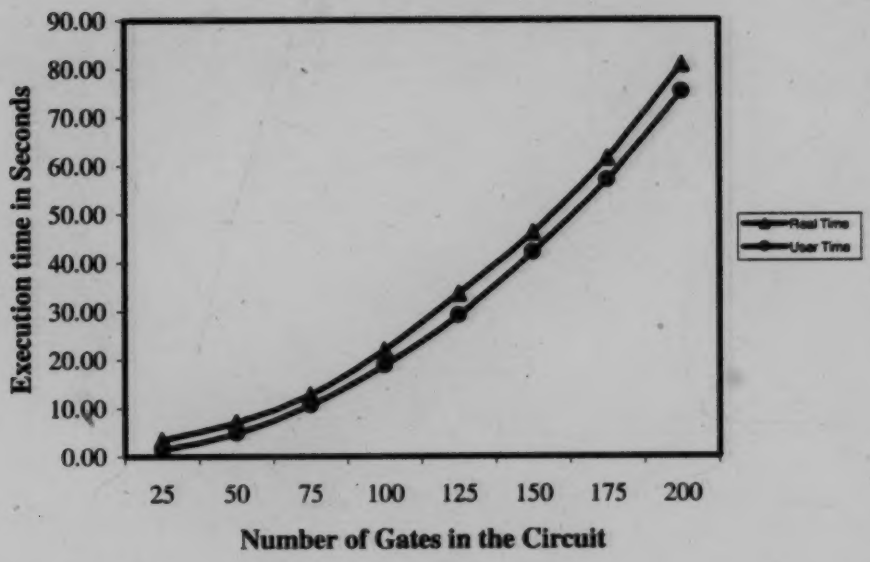


Figure 5.12: Time/Efficiency comparison for an XNOR chain circuit of various lengths

The overall analysis of the circuit, including the voltage and power analysis stages, will be of $O(m^2)$ since the number of transistors is generally greater than the number of nodes in the circuit.

Figure 5.12 shows the real-time results for analysis of a test circuit. The test circuit consists of a chain of varying numbers of T-Cell XNOR gates. The entire time of the program from start to finish is defined as the real-time, and the total time required by the system resources is defined as the user time. The graph is consistent with the proposed theoretical efficiency described above, with tendencies toward $O(m^2)$ for circuits with a large number of gates.

6.0 Conclusions, Findings and Recommendations

6.1 Conclusions

The primary objective of this research was to initiate and develop power dissipation analysis software for CMOS circuits. A subsequent goal was to determine the validity of simulations using the ***power*** software when compared to other methods of simulation analysis.

An educational development version of the software tool was written, compiled and tested for verification. A user supplied netlist and two model library files were imported into the software. The essential data was extracted, used for calculating specific parameters, and then deleted.

The parasitic capacitances were calculated using the netlist file, the model files and approximations using Spice Level 2 techniques. The results of the parasitic capacitances and the netlist supplied interconnect capacitances were used to determine the dynamic energies and powers consumed by the circuit.

The netlist version of the circuit was converted into a single-level hierarchical structure and was used for the remainder of the circuit analysis. The voltages at each node were determined for each transition of the input. Calculations of energy consumption for parasitic and interconnect capacitances were calculated at each node. The average energy consumption for each node was then determined and used to calculate the average dynamic power dissipation. The short-circuit power dissipation contribution was also analyzed and added to the dynamic power dissipation, giving the total power dissipated by the circuit.

All of the results were placed in a Matlab formatted data file along with source code for a Matlab program routine. This provided a graphical user interface to view the energy and power results of the ***power*** program.

The Cadence and Analog Artist programs were used respectively to design and perform initial tests on a variety of circuits. Six simple gate circuits and two complex adders were designed using two distinct standard cell configurations (T-Cell and K-Cell).

The normalized results between the ***power*** program and Analog Artist reveal similar patterns for the test cases. The adder circuits, however, show a slight discrepancy between the various results. This can be attributed to differing analysis methods used by power and Analog Artist, transistor sizing which can cause significant increases in short-circuit power, and voltage supply level requirements for the standard cell configurations (i.e. 3.3V or 5V).

The total power dissipated from the circuit comprises mostly of the dynamic power contribution. The results in Chapter 5 show that the dynamic power dissipated is greater than 80% of the total power. This holds for rise and fall times at input and output of the transistor, being equal, as is the case for the ***power*** program at 1ns. Theory and other research work suggest that the short-circuit contribution will be less than 20% of the total power. The results from ***power*** agree with this theoretical hypothesis [26].

6.2 Limitations

Computer memory could be a critical limitation for the first revisions of the ***power*** program. This problem did not occur in the sample test cases; however, they may occur in larger LSI and VLSI CMOS netlist structures. With significantly large netlist and model data files, memory resources of the computer may reach their limits.

Many of the program subroutines require the use of searches or sorts. They are necessary to import and manipulate the circuit data, calculate capacitances, generate a final circuit structure, calculate energy and power consumption, and export the results to a data file. Linear search/sort procedures were the simplest methods for achieving the results for early program revisions. This comes at a cost of program efficiency, or time to analyze the data, and the amount of memory required for the data lists and data array storage.

The present working version of the ***power*** program limits input test pattern changes to one transition of a single input until the analysis of the circuit is complete. This is known as single transition - forward progression analysis. This method of analysis performs adequately for a small number of inputs to the circuit; however, it may be inefficient for a large number of inputs.

At this time, only CMOS standard gates are considered for analysis. Bipolar, BiCMOS and sequential logic (flip-flop) are not supported at this time.

6.3 Future Revisions

There has been a tremendous increase in the availability of commercial power simulation tools during the past six months. Companies such as Cadence and Mentor Graphics have seen a niche for this type of software in addition to their existing CAD products. The ***power*** software developed for this research is in its infancy stage, and requires some major and minor alterations if future work is to continue and the software is to compete with commercial tools.

The results obtained and presented in Chapter 5 are comparable with simulated and theoretical results from other methods. However, the real-time measurement testing of circuits designed and simulated is required. This is necessary for the verification of the results. Designs need to be fabricated and tested for functionality prior to measuring the devices in real-time. This is costly and is a major factor if research into the ***power*** program is to continue.

"C" is a stalwart programming language, however object-oriented programming is becoming the preferred method of generating software. The rewriting of the ***power*** program in "C++" would result in significant portability and efficiency improvements of this tool. This would be one method to let the software become multi-platformed, whereas it is now Unix dependant.

Revision of the software is required to improve efficiency and memory allocation constraints. Most of the search algorithms used in the current program are linear. Therefore, more efficient algorithms need to be considered.

Multiple transition - forward progression analysis should be considered for future revisions of the software. This should improve efficiency of the circuit analysis, and would be more convenient for larger circuit designs and designs using sequential synchronous and asynchronous devices.

At present, only CMOS combinational logic circuits can be analyzed. Future revisions to the ***power*** program will include an improved analysis structure for netlist files. This will include the addition of bipolar and BiCMOS transistor models. Methods for the analysis of data storage elements such as sequential logic flip-flops and latches, will also be addressed.

Changing the programming language from "C" to "C++" will provide a new method of graphically viewing the data. A Matlab oriented program is appended to the data file, thus requiring Matlab software application to view the data. Future work would make a transition to OpenGL programming, and "C++", for the graphical user interface portion of the program.

Bibliography

TEXTBOOKS

[1] A. Bellaovar, M.I. Elmasry, "Low Power Digital VLSI Design: Circuits and Systems," Kluwer Academic Publishers, MA 1995.

[2] N.H.E. West, K. Eshraghiam, "Principles of CMOS VLSI Design: A Systems Perspective 2nd Edition," Addison Wesley, 1994.

[3] J.M. Rabaey, "Digital Integrated Circuits: A Design Perspective," Prentice Hall, 1996.

[4] D. Foty, "MOSFET Modeling with Spice: Principles and Practice," Prentice Hall, 1997.

[5] J.M. Rabaey, M. Pedram, "Low Power Design Methodologies," Kluwer Academic Publishers, 1996.

[6] D. Gilly, O'Reilly & Associates, "UNIX in a Nutshell," O'Reilly & Associates Inc., 1992.

[7] R.N. Horspool, "The Berkeley UNIX Environment, 2nd Edition," Prentice Hall, 1992.

[8] B.W. Kernigham, D.M. Ritchie, "The C Programming Language, 2nd Edition," Prentice Hall, 1995.

[9] M.A. Weiss, "Efficient C Programming Language," Prentice Hall, 1995.

[10] The Math Works Inc., "Matlab Reference and User's Guide," The Math Works, 1992.

PAPERS

[11] M. Horowitz, T. Indermaur, R. Gonzalez, "Low-Power Digital Design," 1994 IEEE Symposium on Low Power Electronics, pp. 8 – 11.

[12] T. Kuroda, T. Sakvrai, "Overview of Low-Power ULSI Circuits Techniques," IEICE Transactions on Electronics, Special Issue of Low-Voltage, Low-Power Integrated Circuits, Vol. E78-C, No. 4, April 1995, pp. 334-344.

[13] G. Frenkil, "Power Dissipation of CMOS ASICs," IEEE Proceedings 4th Annual International ASIC Conference and Exhibit, 1991, pp. T3-1.1 – T3-1.5.

[14] A. Shah, P. Young, "MOS Technology: Trends and Challenges in the ULSI Era," IEEE Proceedings 20th International Conference on Microelectronics (MIEL '95), Vol. 1, September 1995, pp. 3-9.

[15] S. Devadas, S. Malik "A Survey of Optimization Techniques Targeting Low Power VLSI Circuits," 32nd ACM/IEEE Design Automation Conference 1995.

[16] Z. Chen, J. Shott, J. Burr, J.D. Plummer, "CMOS Technology Scaling for Low-Voltage Low-Power Applications," 1994 IEEE Symposium on Low-Power Electronics, pp. 56-57.

[17] J.A. Segura, M. Roca, D. Mateo, A. Rubio, "An Approach to Dynamic Power Consumption Current Testing CMOS ICs," IEEE 1995.

[18] F. Najm, " Feedback, Correlation, and Delay Concerns in the Power Estimation of VLSI Circuits," 32nd ACM/IEEE Design Automation Conference 1995.

[19] G. Burd, "Low-Power CMOS Library Design Methodology," Masters of Science, Electrical Engineering and Computer Science Thesis, University of California, Berkeley.

[20] P. Gutwin, "A Web Document: Low Power CMOS," www.emba.uvm.edu/~pgutwin/lowpower/lowpower.html.

[21] R. Zimmermann, "A Web Document: "Power Calculator for Compass." Integrated Systems Laboratory, 1997, www.stud.ee.ethz.ch/~Zimm/powercalc/powercalc.html.

[22] A. Ashkinazy, D. Edwards, C. Farnsworth, G.Gendel, S. Sikand, "Tools for Validating Asynchronous Digital Circuits."

[23] EPIC, "Technical White Paper on Power," EPIC Design Technology, 1996

[24] EPIC, "Power Mill Product Brief," EPIC Design Technology, 1996.

[25] M. Margala, A 4-2 Compressor Circuit: Design for Doctoral Research, Ph.D., Electrical and Computer Engineering, University of Alberta" Paper status unavailable.

[26] H.J.M. Veendrick, "Short-Circuit Dissipation of Static CMOS Circuitry and its Impact on the Design of Buffer Circuits," IEEE Journal of Solid-State Circuits, Vol. SC-19, No. 4, August 1984.

[27] A. Vinnakota, "Monitoring Power Dissipation for Fault Detection," Journal of Electronic Testing: Theory and Application 11, pp. 173-181, Kluwer Academic Publishers, 1997.

Appendix A.1 PMOS Model Library File

The following is a portion of the PMOS library file. The file consists of 0.8µm, 1.2µm, 2µm, 5µm, and 10µm technologies in its entirety.

```
* Canadian Microelectronics Corporation
* BiCMOS Design Kit V1.0 for Cadence Analog Artist
* August 23, 1995
*
* 0.8-micron BiCMOS SPECTRE library for MOSFET models
*
* Dec 6/94, Hsu Ho, CMC
*          - ETA and KAPPA were tuned for devices of 4.0um or greater
*            channel lengths to effect better modelling for MOS output
*            conductance.
*
* You should have the following options set in order to supress the
* printing of the model parameters, to make the nominal temperature
* compatible with SPICE, and to make SPECTRE use the same device models
* as SPICE 2G.6:
*
*  .OPTIONS NOMOD TNOM=27 DCAP=1
*
*********************************************************************
*
* THE FOLLOWING BiCMOS MODELS ARE AVAILABLE:
*
* P-CHANNEL MOSFET:
* SEMI-SCALABLE DEVICE NAME: MPCH
* CALLING CARD: M_ D G S B MP___ L=X W=X NRD=X NRS=X AD=X AS=X PD=X PS=X
*
#ifdef TYPICAL
*
*  SEMI-SCALABLE PMOS PARAMETER LIBRARY
*
*  P-CHANNEL MOSFET  GATE LENGTH = 0.8 um  GATE WIDTH = any
.MODEL  MPCH_0P8  PMOS
+    XL=0.00          XW=0.00
+  //LMAX=1.05E-6  //LMIN=0.6E-6
+  //WMAX=500E-6  //WMIN=1.4E-6
+  LEVEL=3  //ACM=2
*  Long channel model parameters
+    VTO=-0.902       UO=154          TOX=17.52E-9
+    NSUB=3.149E16    NFS=760E9       DELTA=0.295
+    THETA=128E-3     WD=100E-9
*  Short channel model parameters
+    LD=1.000E-9      LDIF=999.0E-9   XJ=308E-9
+    VMAX=277.3E3     ETA=79.53E-3    KAPPA=9.56
+    RS=1.200E3       RSH=0.0         HDIF=1.2E-6
+    RD=1.200E3
*  Temperature model parameters
+    TLEV=1           TCV=973.4E-6    BEX=-1.100
+    TRS=1.000E-9     TRD=1.000E-9
*  Diode model parameters
+  JS=5.0E-4  //JSW=5.5E-10
+    CJ=450.0E-6      MJ=0.61
+    CJSW=220.0E-12   MJSW=0.26
+  //CJGATE=820.0E-12
+  PB=0.921  PBSW=0.921
+    CGSO=214.8E-12   CGBO=568.3E-12
+    CGDO=214.8E-12
*
*
```

```
* Junction Temperature model parameters
+    TLEVC=1
+    PTA=1.39E-3        PTP=4.55E-3
+    CTA=607E-9         CTP=160E-15
* Noise model parameters
+ AF=0.95  KF=250.0E-27  //NLEV=2
*
#endif// TYPICAL
```

Appendix A.2 NMOS Library File

The following is a portion of the NMOS library file. The file consists of 0.8µm, 1.2µm, 2µm, 5µm, and 10µm technologies, in its entirety.

```
* Canadian Microelectronics Corporation
*
* BiCMOS Design Kit V1.0 for Cadence Analog Artist
* August 23, 1995
*
* 0.8-micron BiCMOS SPECTRE library for MOSFET models
*
* Dec 6/94 Hsu Ho, CMC
*          - ETA and KAPPA were tuned for devices of 4.0um or greater
*      channel lengths to effect better modelling for MOS output
*      conductance.
*
* You should have the following options set in order to supress the
* printing of the model parameters, to make the nominal temperature
* compatible with SPICE, and to make SPECTRE use the same device models
* as SPICE 2G.6:
*
*  .OPTIONS NOMOD TNOM=27 DCAP=1
*
*********************************************************************
*
* THE FOLLOWING BiCMOS MODELS ARE AVAILABLE:
*
* N-CHANNEL MOSFET:
* SEMI-SCALABLE DEVICE NAME: MNCH
* CALLING CARD: M_ D G S B MN___ L=X W=X NRD=X NRS=X AD=X AS=X PD=X PS=X
*
*
#ifdef TYPICAL
*
*  SEMI-SCALABLE NMOS PARAMETER LIBRARY
*
*  N-CHANNEL MOSFET  GATE LENGTH = 0.8 um  GATE WIDTH = any
.MODEL  MNCH_0P8  NMOS
+    XL=0.00          XW=0.00
+  //LMAX=1.0E-6  //LMIN=0.6E-6
+  //WMAX=500E-6  //WMIN=1.4E-6
+  LEVEL=3  //ACM=2
*  Long channel model parameters
+    VTO=0.8115       UO=475           TOX=17.52E-9
+    NSUB=3.618E16    NFS=734.5E9      DELTA=1.0529
+    THETA=52.45E-3   WD=45.0E-9
*  Short channel model parameters
+    LD=72.67E-9      LDIF=924.3E-9    XJ=160.4E-9
+    VMAX=146.5E3     ETA=36.06E-3     KAPPA=1e-12
+    RS=1.076E3       RSH=0.0          HDIF=1.2E-6
+    RD=1.076E3
*  Temperature model parameters
+    TLEV=1           TCV=980.8E-6     BEX=-1.650
+    TRS=-1.000E-9    TRD=-1.000E-9
*  Diode model parameters
+  JS=5.0E-4  //JSW=5.5E-10
+    CJ=260.0E-6      MJ=0.46
+    CJSW=280.0E-12   MJSW=0.20
+  //CJGATE=930.0E-12
+  PB=0.925  PBSW=0.925
+    CGSO=288.4E-12   CGBO=568.3E-12
+    CGDO=288.4E-12
```

```
* Junction Temperature model parameters
+   TLEVC=1
+   PTA=2.90E-3       PTP=8.24E-3
+   CTA=353E-9        CTP=3.18E-15
* Noise model parameters
+ AF=0.85  KF=1.5E-24  //NLEV=2
*
#endif// TYPICAL
```

Appendix A.3 NPN Library File

The following is a portion of the NPN library file. The file is extremely large with various configurations of NPN BiCMOS transistors. Therefore only one example is shown since the ***power*** program does not require use of this file.

```
* Canadian Microelectronics Corporation
* BiCMOS Design Kit V1.0 for Cadence Analog Artist
* August 23, 1995
*
* 0.8-micron BiCMOS SPECTRE library for 5V bipolar models
*
* Release:    1D02.5
* Date:       January 11, 1994
*
* You should have the following options set in order to supress the
* printing of the model parameters, to make the nominal temperature
* compatible with SPICE, and to make SPECTRE use the same device models
* as SPICE 2G.6:
*
*   .OPTIONS NOMOD TNOM=27 DCAP=1
*
*********************************************************************
*
* The following BiCMOS models are available:
*
* VERTICAL NPN:
* DEVICE NAMES: NN51111X NN52111X NN52112X NN52114X NN53215X NN52122X
*          NN52124X NN532215 NN564230 NN596245 NN532315 NN564330
*          NN596345 NN594115 NN51X11X NN52224X NN52225X
*          NN52F11X NN52F24X NN52F25X NN53F215 NN56F230 NN59F245
*          NN564B30 NN596B45 NN52F12X NN52F14X NN521A1X
*          NN5CF3C6 NN5C212X
* CALLING CARD: Q_ C B E S NN5____
*
*
#ifdef  TYPICAL
*
*  BIPOLAR PARAMETER LIBRARY
**
*  1X NPN WITH SINGLE BASE CONTACT CONTACTED WITH METAL 1
.MODEL NN51111X NPN
+      IS= 3.10E-18      BF= 1.03E+02      NF= 1.00E+00
+     VAF= 6.00E+01      IKF= 5.36E-03      ISE= 4.57E-18
+      NE= 1.50E+00                  ISS= 1.00E-17
+      BR= 1.00E+00      NR= 1.00E+00      VAR= 5.50E+00
+ NC=2.00E+00 ISC=0.00E+00 //IKR=0.00E+00
+ RB=9.66E+02 RBM=9.66E+02 //IRB=0.00E+00
+      RE= 1.96E+01      RC= 6.28E+01
+     CJE= 1.67E-14      VJE= 8.00E-01      MJE= 2.67E-01
+      TF= 1.25E-11     XTF= 7.36E+02      VTF= 1.31E+00
+     ITF= 9.38E-02     PTF= 4.00E+01
+     CJC= 2.02E-14      VJC= 7.10E-01      MJC= 3.97E-01
+    XCJC= 5.10E-01       TR= 4.00E-09
+     CJS= 3.44E-14      VJS= 3.70E-01      MJS= 1.34E-01
+     XTB= 1.62E+00       EG= 1.11E+00      XTI= 5.82E+00
+      KF= 4.27E-09       AF= 2.07E+00       FC= 8.00E-01
*
#endif// TYPICAL
```

Appendix B.1 Typedefs.h Header File Information

This file defines the data structures used in the initial stages of the ***power*** program. There are two naming schemes involved with this file. The first defines the netlist data linked list structure. The second structure defines the transistor model data linked list.

Structure	Definition
struct cct (CCT):	This is the structure for the Sub-circuit linked list. References are made to other structures.
struct cap (CAP):	This is the structure for Capacitor information. It is used as a stand alone structure, or part of the CCT structure as a sub-linked list.
struct mos (MOS):	This is the structure for Mosfet information. It is used as a stand alone structure for a linked list, or part of the CCT structure as a sub-linked list.
struct que (QUE):	This is the structure for Transistor information. It is used as a stand alone structure for a linked list, or part of the CCT structure as a sub-linked list.
struct res (RES):	This is the structure for Resistor information. It is used as a stand alone structure for a linked list, or part of the CCT structure as a sub-linked list.
struct sub (SUB):	This is the structure for the Sub-circuit reference information.
struct vol (VOL):	This is the structure for the Voltage reference information.

Table B.1: Netlist linked list data structures.

Figures B.1 to B.7 show the contents and definitions of each netlist linked list styles.

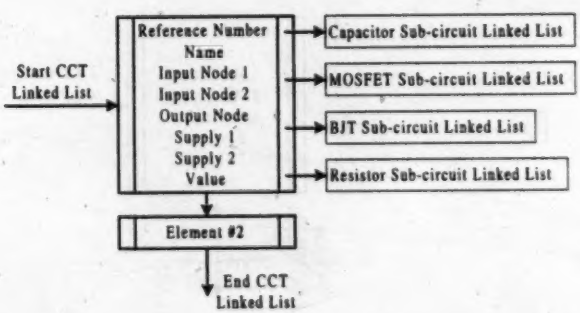


Figure B.1: CCT (sub-circuit data) linked list structure.

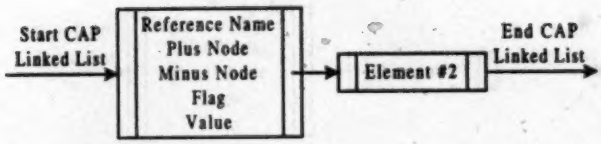


Figure B.2: CAP (capacitor data) linked list structure.

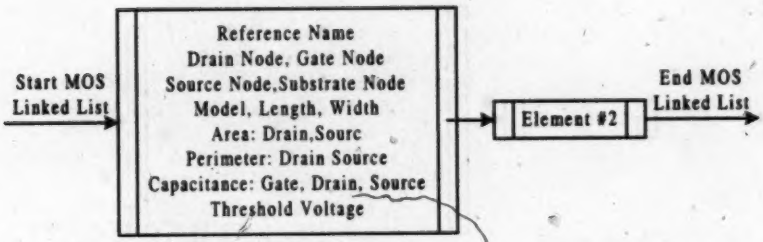


Figure B.3: MOS (MOSFET data) linked list structure.

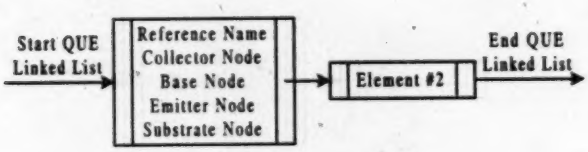


Figure B.4: QUE (transistor data) linked list structure.

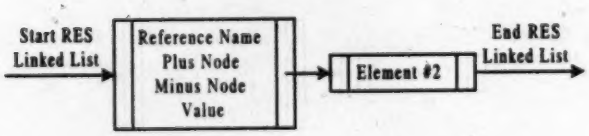


Figure B.5: RES (resistor data) linked list structure.

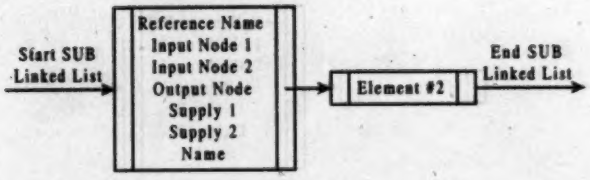


Figure B.6: SUB (sub-circuit reference data) linked list structure.

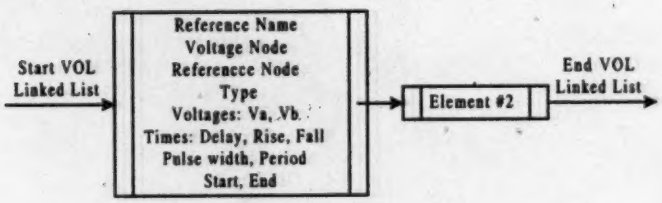


Figure B.7: VOL (voltage data) linked list structure.

This is the second structure defines the transistor model data linked list

Structure	Definition
struct model (MODEL):	This is the structure for the Model reference information. In this structure, references are made to other structures.
struct long_param (LONG):	This is the structure for the long parameters of the mosfet model.
struct short_param (SHORT):	This is the structure for the short parameters of the mosfet model.
struct temperature_param (TEMP):	This is the structure for the temperature parameters of the mosfet model.
struct diode_param (DIODE):	This is the structure for the diode parameters of the mosfet model.
struct junction_param (JUNCT):	This is the structure for the junction parameters of the mosfet model.
struct noise_param (NOISE):	This is the structure for the noise parameters of the mosfet model.

Table B.2: Model linked list data structures.

Figures B.8 to B.14 show the contents and definitions of each model linked list and data parameter styles.

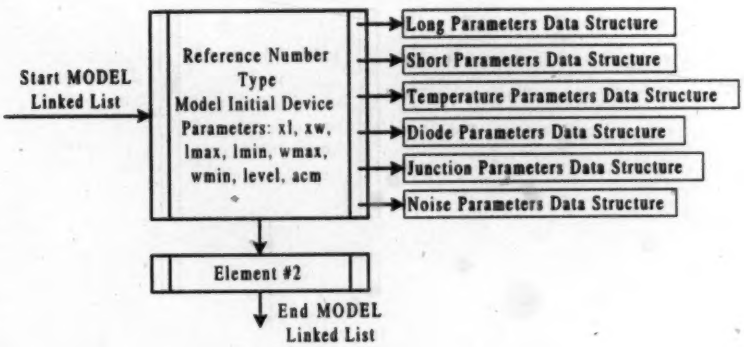


Figure B.8: MODEL (model library) linked list structure.

Figure B.9: LONG (long parameters) data structure.

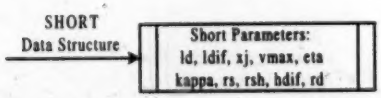


Figure B.10: SHORT (short parameters) data structure.

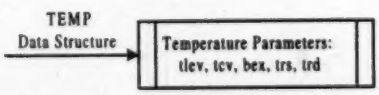


Figure B.11: TEMP (temperature parameters) data structure.

Figure B.12: DIODE (diode parameters) data structure.

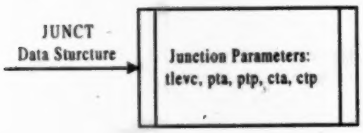


Figure B.13: JUNCT (junction parameters) data structure.

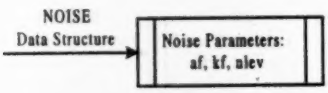


Figure B.14: NOISE (noise parameters) data structure.

Appendix B.2 Combine.h Header File Information

This file defines the data structures used in the final stages of the ***power*** program. This list contains the entire flat, single level hierarchical circuit representation.

Structure	Definition
struct nodes (NODES):	This is the structure for the Node information. It is used as a stand alone linked list structure to describe all of the nodes in the CUT.
struct clist (CLIST):	This is the structure for Capacitor information. It is used as a stand alone linked list structure for describing all capacitors in the final circuit netlist.
struct mlist (MLIST):	This is the structure for Mosfet information. It is used as a stand alone linked list structure for describing all mosfets in the final circuit netlist.
struct tlist (TLIST):	This is the structure for Transistor information. It is used as a stand alone linked list structure for describing all transistors in the final circuit netlist
struct rlist (RLIST):	This is the structure for Resistor information. It is used as a stand alone linked list structure for describing all resistors in the final circuit netlist.

Table B.3: Final circuit linked list data structures.

Figures B.15 to B.19 show the contents and definitions of each final circuit linked list data styles.

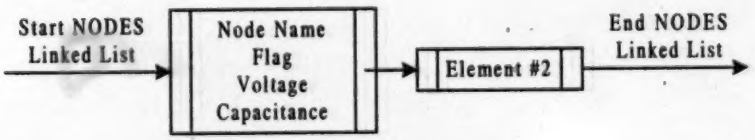


Figure B.15: NODES (node data) linked list structure.

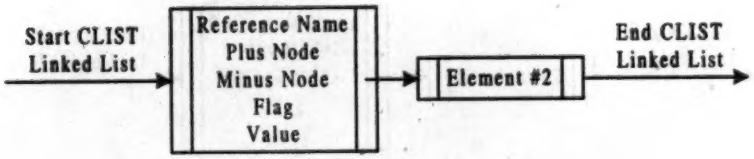


Figure B.16: CLIST (capacitor data) linked list structure.

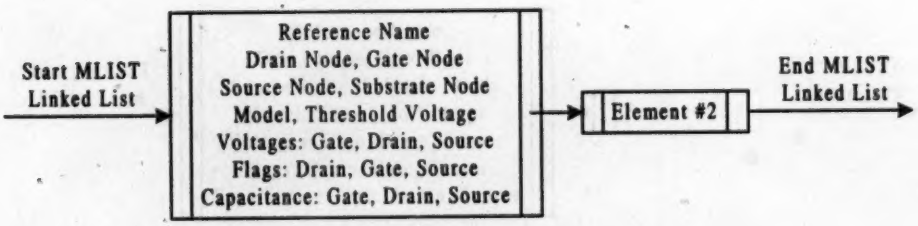


Figure B.17: MLIST (MOSFET data) linked list structure.

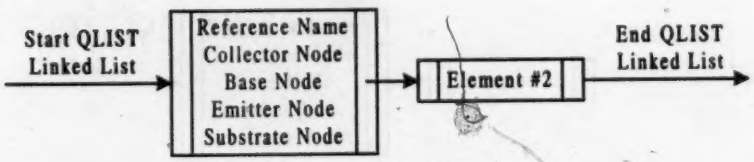


Figure B.18: QLIST (transistor data) linked list structure.

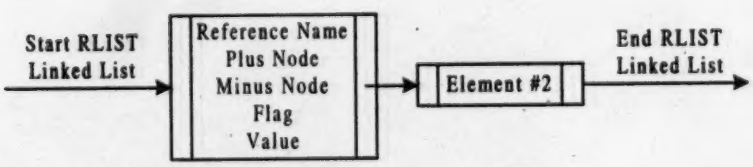


Figure B.19: RLIST (resistor data) linked list structure.

Appendix C.1 Program Block Flow Diagram

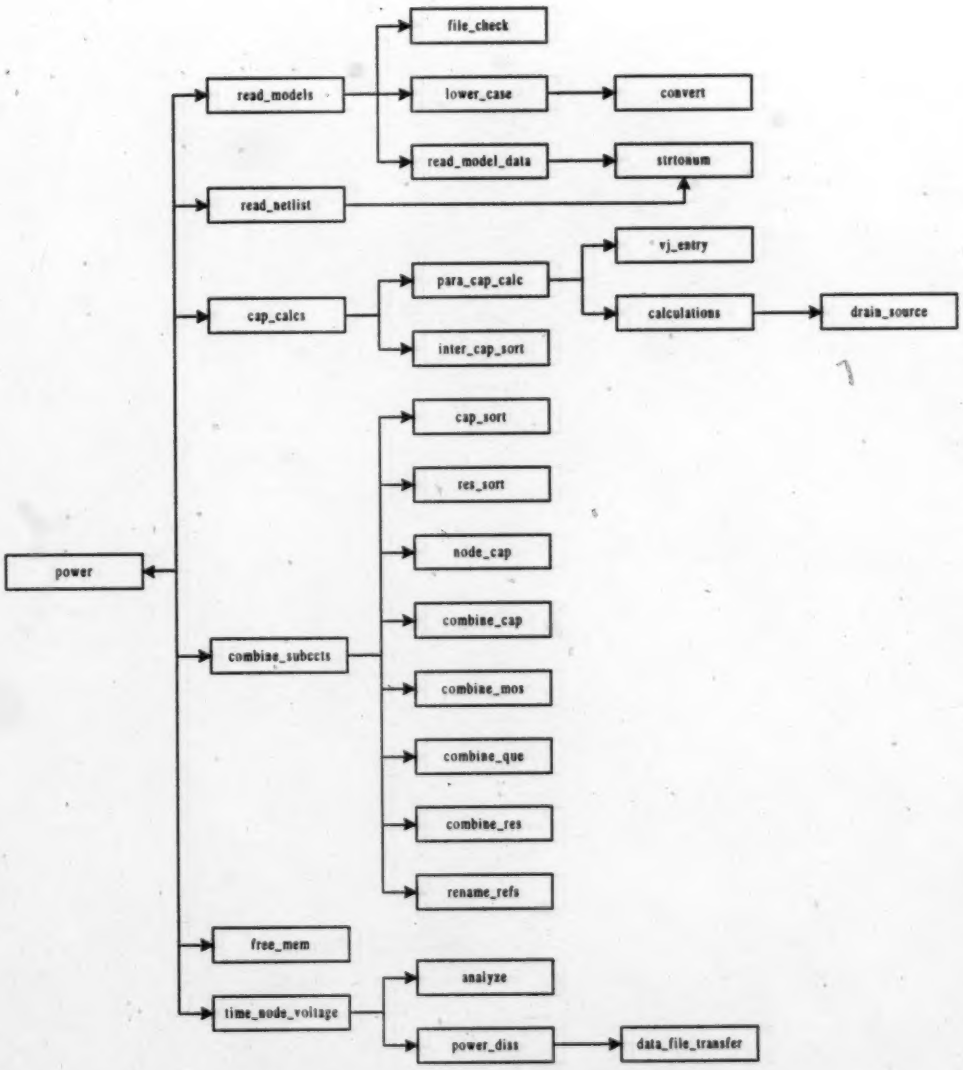


Figure C.1: ***power*** **program main block diagram.**

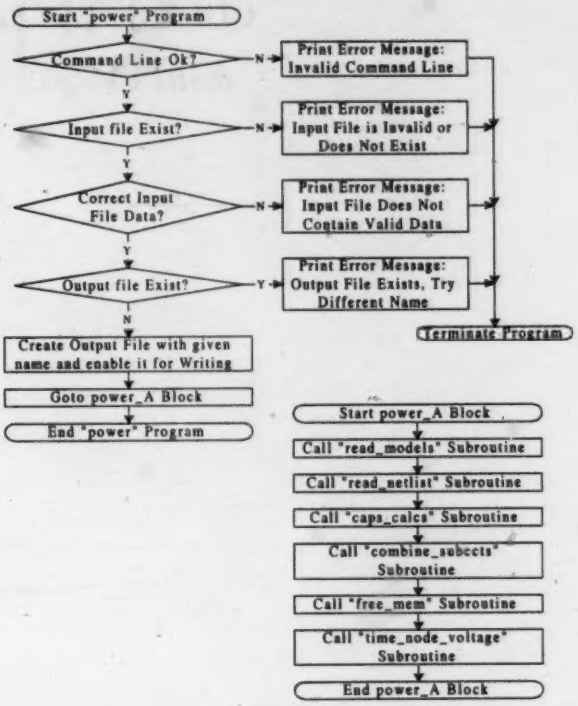


Figure C.2: Flowchart for ***power*** main program routine.

Appendix C.2 Level – One Program Flowcharts

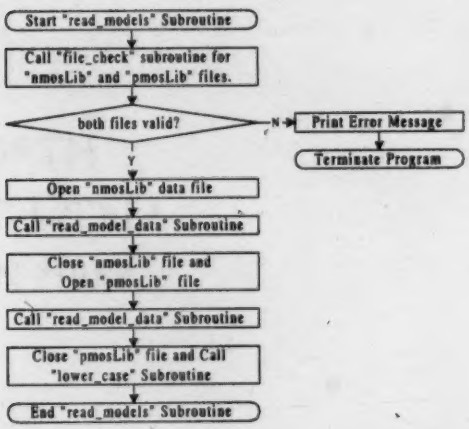


Figure C.3: Flowchart for "read_models" subroutine.

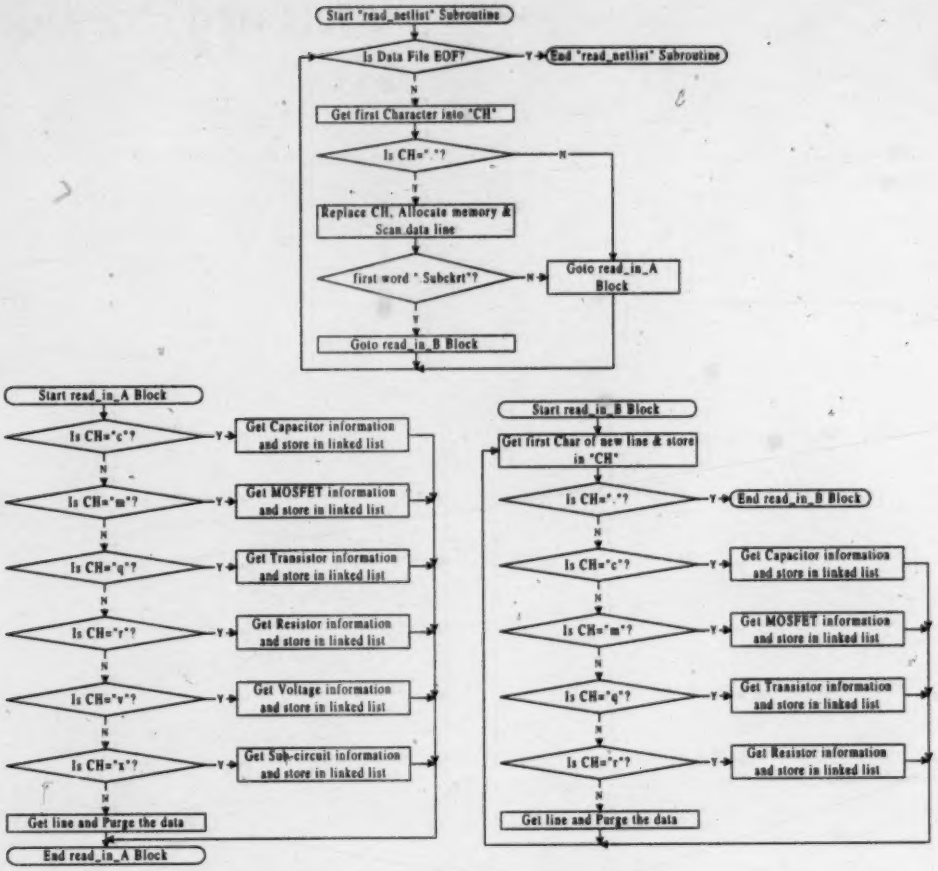

Start "read_netlist" Subroutine
Is Data File EOF?
End "read_netlist" Subroutine
Get first Character into "CH"
Is CH=" "?
Replace CH, Allocate memory & Scan data line
first word * Subckt*?
Goto read_in_A Block
Goto read_in_B Block
Start read_in_A Block
Is CH="c"?
Get Capacitor information and store in linked list
Is CH="m"?
Get MOSFET information and store in linked list
Is CH="q"?
Get Transistor information and store in linked list
Is CH="r"?
Get Resistor information and store in linked list
Is CH="v"?
Get Voltage information and store in linked list
Is CH="x"?
Get Sub-circuit information and store in linked list
Get line and Purge the data
End read_in_A Block
Start read_in_B Block
Get first Char of new line & store in "CH"
Is CH="."?
End read_in_B Block
Is CH="c"?
Get Capacitor information and store in linked list
Is CH="m"?
Get MOSFET information and store in linked list
Is CH="q"?
Get Transistor information and store in linked list
Is CH="r"?
Get Resistor information and store in linked list
Get line and Purge the data

Figure C.4: Flowchart for "read_netlist" subroutine.

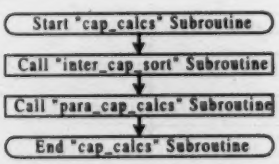


Figure C.5: Flowchart for "cap_calcs" subroutine.

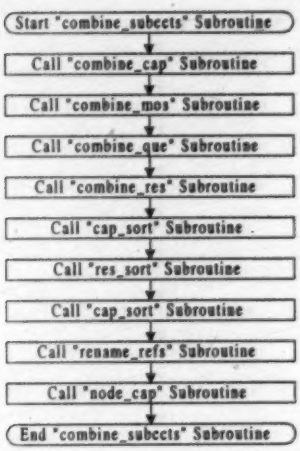


Figure C.6: Flowchart for "combine_subccts" subroutine.

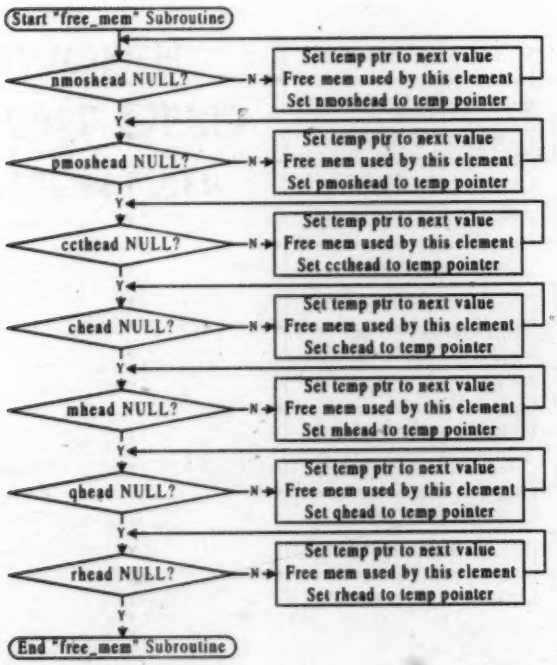


Figure C.7: Flowchart for "free_mem" subroutine.

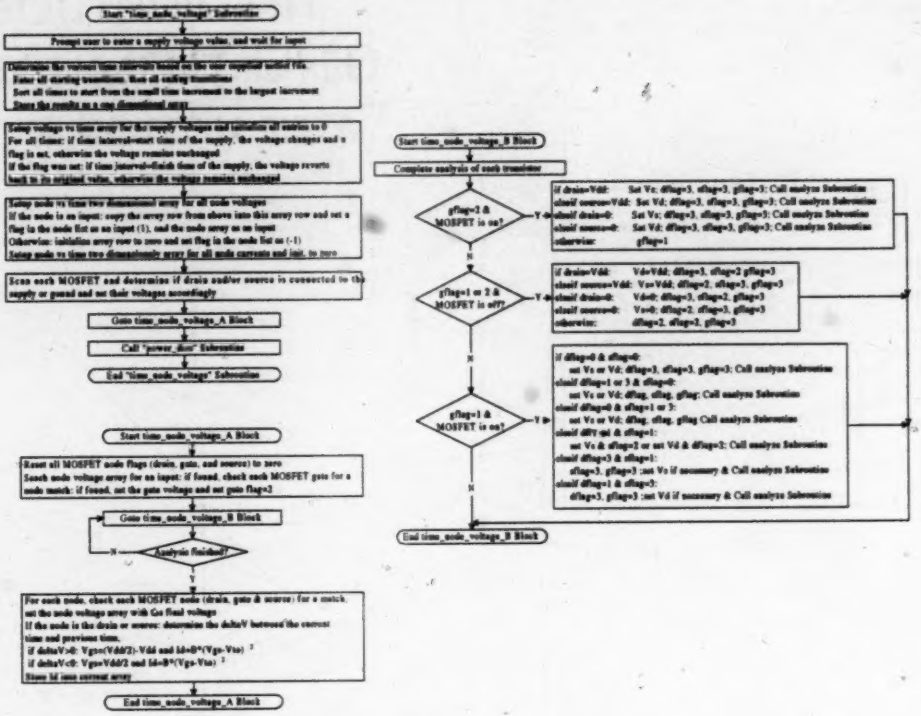

Start "time_node_voltage" Subroutine
Prompt user to enter a supply voltage value, and wait for input
Setup voltage vs time array for the supply voltages and initialize all entries to 0
Setup node vs time two dimensional array for all node voltages
Setup node vs time two dimensionally array for all node currents and init. to zero
Scan each MOSFET and determine if drain and/or source is connected to the supply or ground and set their voltages accordingly
Goto time_node_voltage_A Block
Call "power_dissi" Subroutine
End "time_node_voltage" Subroutine
Start time_node_voltage_A Block
Reset all MOSFET node flags (drain, gate, and source) to zero
Goto time_node_voltage_B Block
Analysis finished?
N
Y
End time_node_voltage_A Block
Start time_node_voltage_B Block
Complete analysis of each transistor
gflag=2 & MOSFET is on?
gflag=1 or 2 & MOSFET is off?
gflag=1 & MOSFET is on?
Y
N
End time_node_voltage_B Block

Figure C.8: Flowchart for "time_node_voltage" subroutine.

Appendix C.3 Level – Two Program Flowcharts

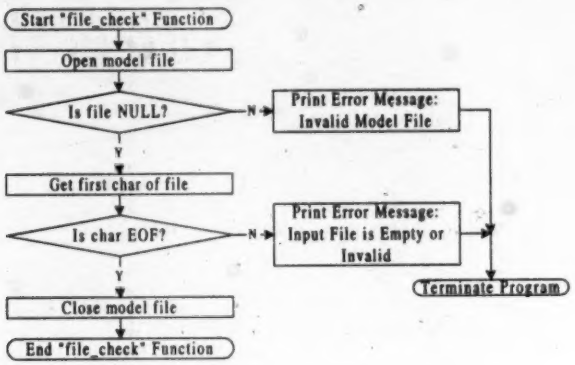


Figure C.9: Flowchart for "file_check" function.

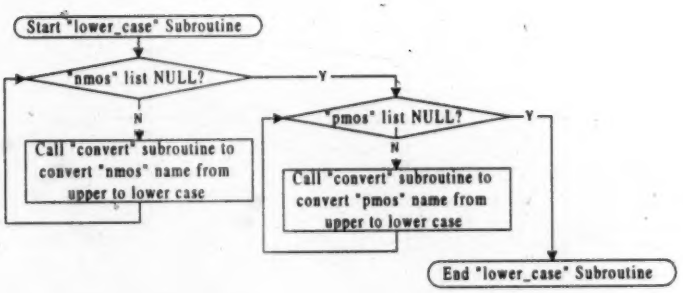

Start "lower_case" Subroutine
"nmos" list NULL?
Y
N
"pmos" list NULL?
Y
N
Call "convert" subroutine to convert "nmos" name from upper to lower case
Call "convert" subroutine to convert "pmos" name from upper to lower case
End "lower_case" Subroutine

Figure C. 10: Flowchart for "lower_case" subroutine.

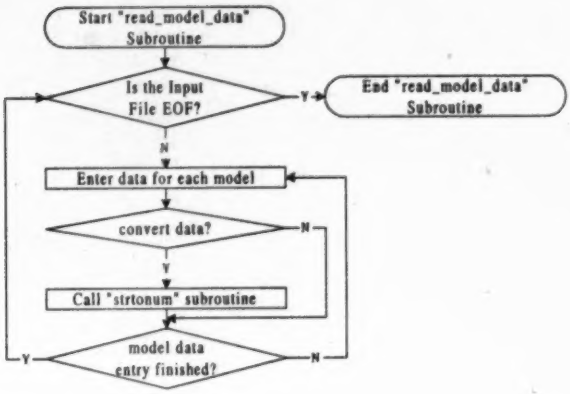


Figure C. 11: Flowchart for "read_model_data" subroutine.

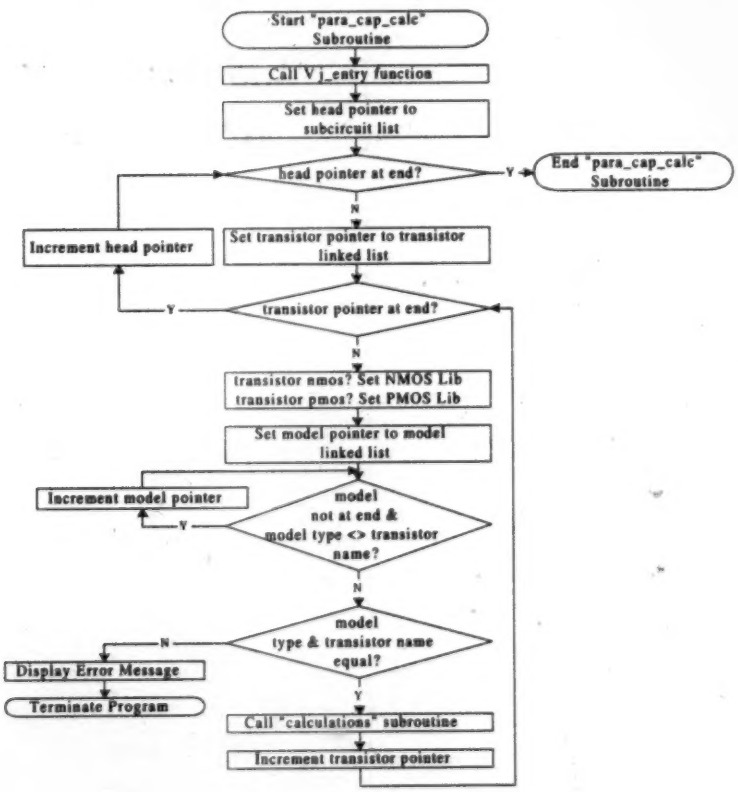


Figure C. 12: Flowchart for "para_cap_calc" subroutine.

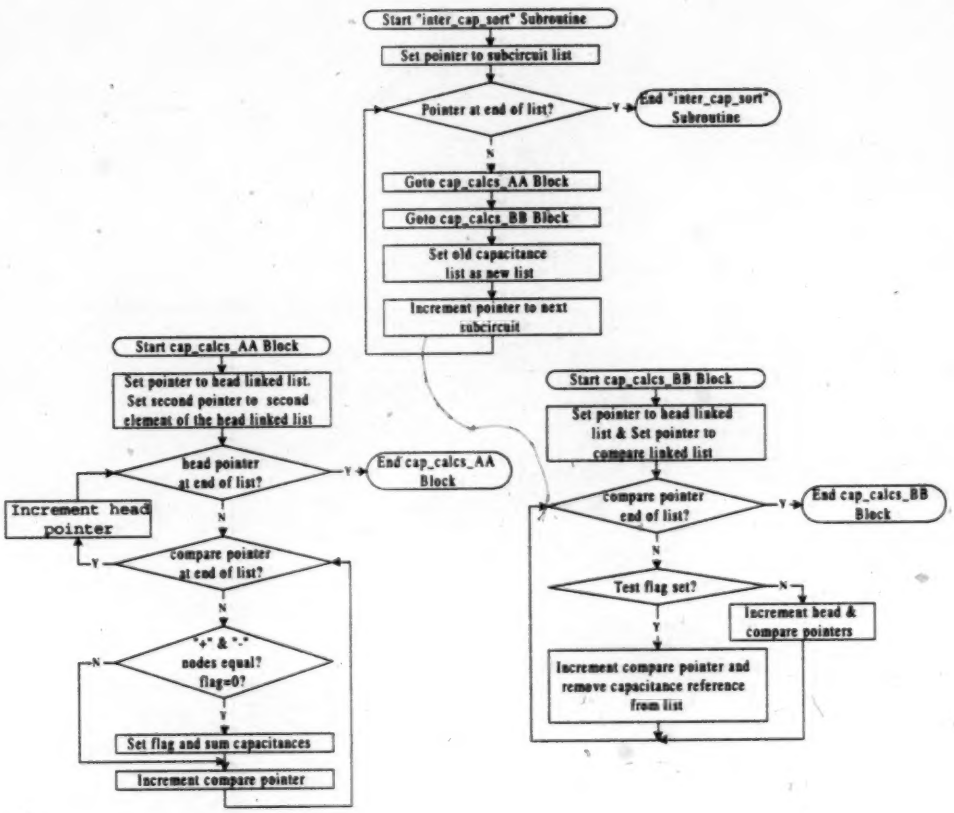


Figure C.13: Flowchart for "inter_cap_sort" subroutine.

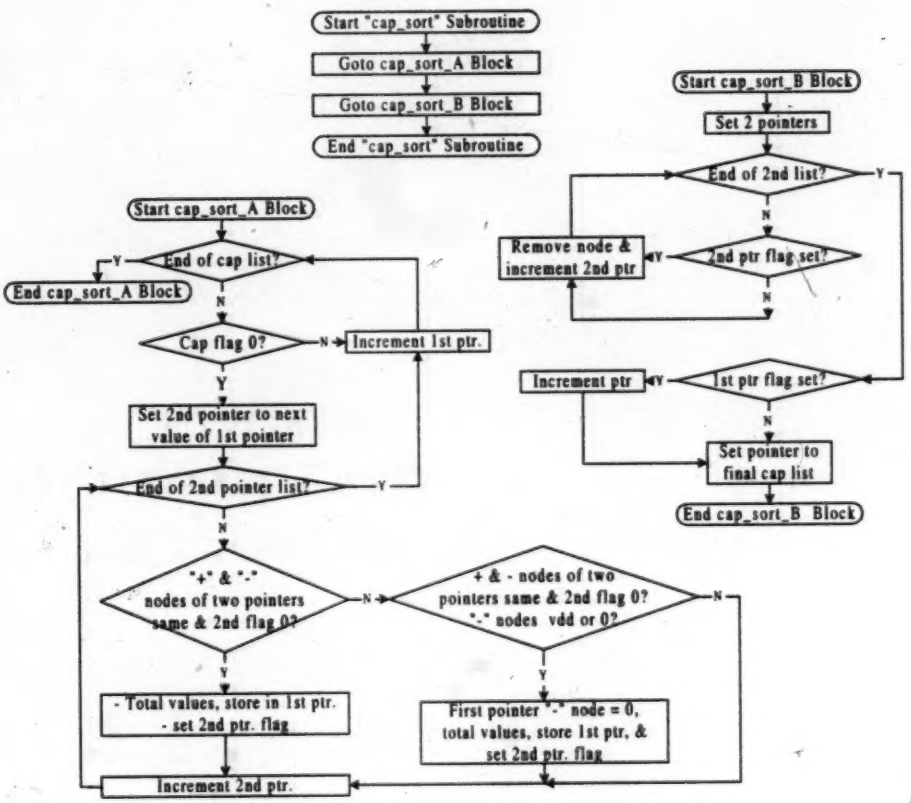


Figure C.14: Flowchart for "cap_sort" subroutine.

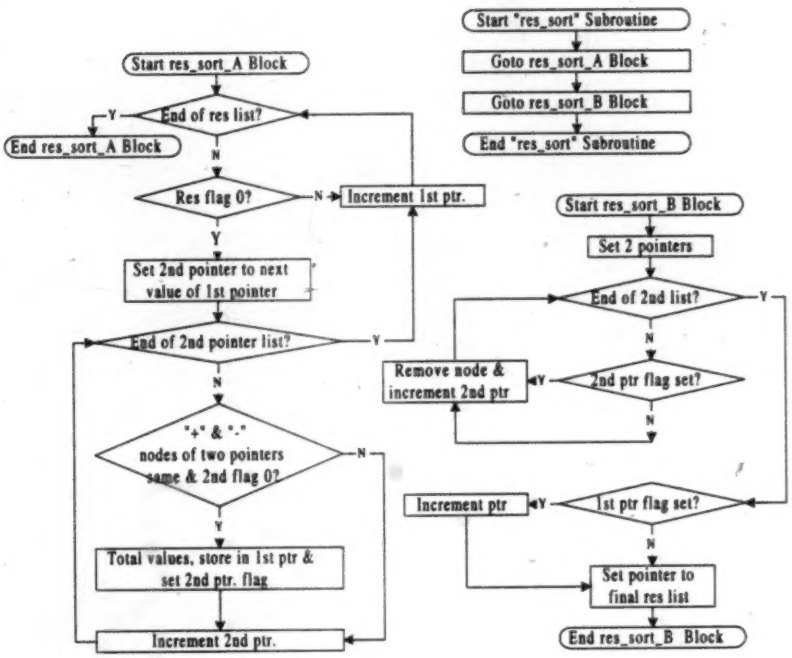


Figure C.15: Flowchart for "res_sort" subroutine.

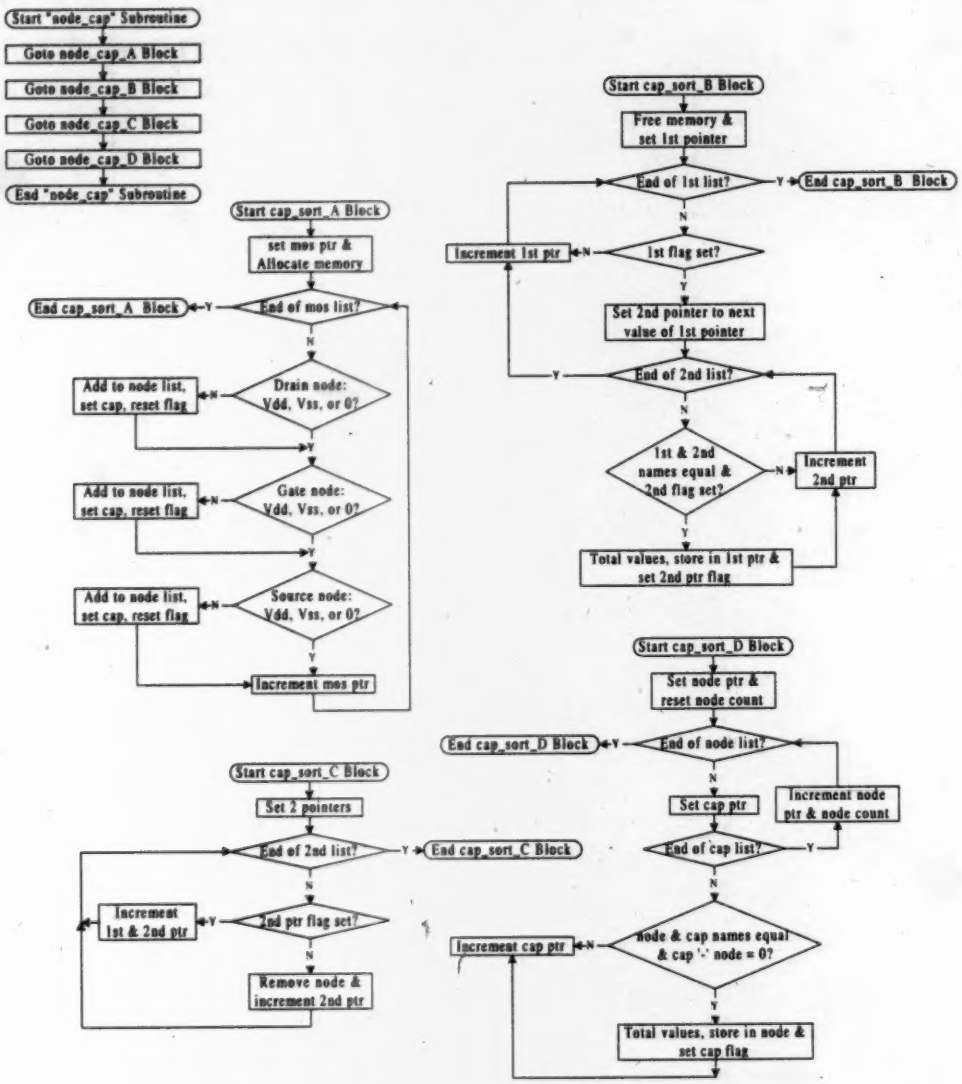

Start "node_cap" Subroutine
Goto node_cap_A Block
Goto node_cap_B Block
Goto node_cap_C Block
Goto node_cap_D Block
End "node_cap" Subroutine
Start cap_sort_A Block
set mos ptr & Allocate memory
End cap_sort_A Block
End of mos list?
Add to node list, set cap, reset flag
Drain node: Vdd, Vss, or 0?
Add to node list, set cap, reset flag
Gate node: Vdd, Vss, or 0?
Add to node list, set cap, reset flag
Source node: Vdd, Vss, or 0?
Increment mos ptr
Start cap_sort_B Block
Free memory & set 1st pointer
End of 1st list?
End cap_sort_B Block
Increment 1st ptr
1st flag set?
Set 2nd pointer to next value of 1st pointer
End of 2nd list?
1st & 2nd names equal & 2nd flag set?
Increment 2nd ptr
Total values, store in 1st ptr & set 2nd ptr flag
Start cap_sort_C Block
Set 2 pointers
End of 2nd list?
End cap_sort_C Block
Increment 1st & 2nd ptr
2nd ptr flag set?
Remove node & increment 2nd ptr
Start cap_sort_D Block
Set node ptr & reset node count
End cap_sort_D Block
End of node list?
Set cap ptr
Increment node ptr & node count
End of cap list?
Increment cap ptr
Node & cap names equal & cap '-' node = 0?
Total values, store in node & set cap flag

Figure C.16: Flowchart for "node_cap" subroutine.

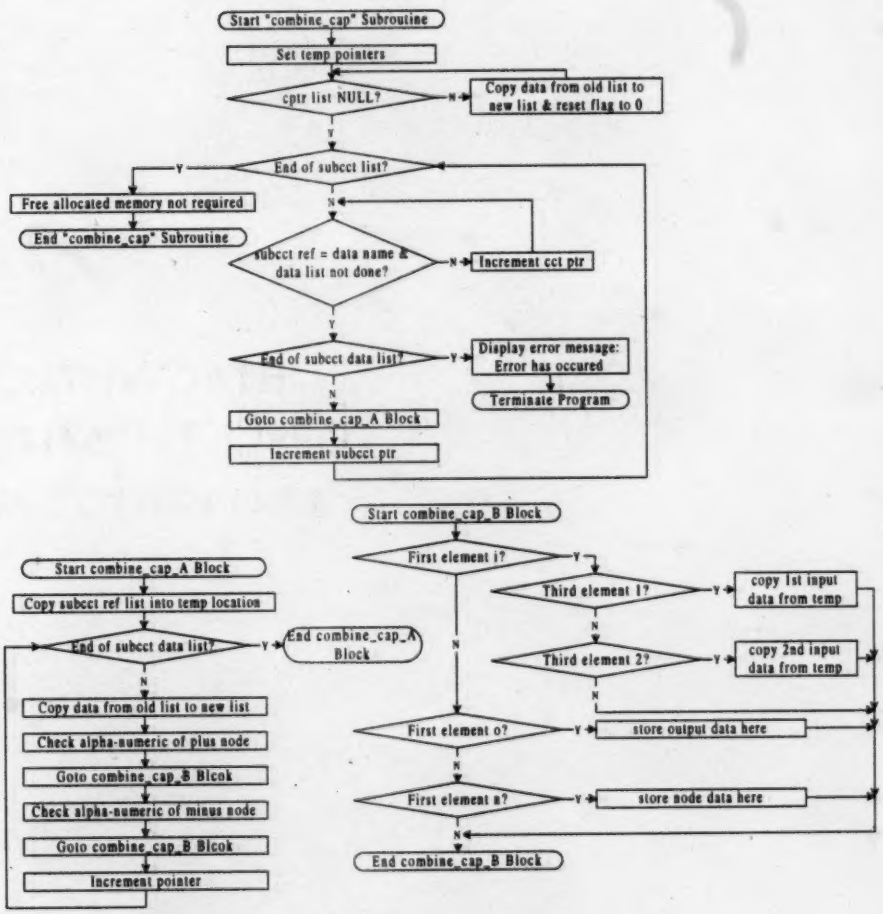


Figure C. 17: Flowchart for "combine_cap" subroutine.

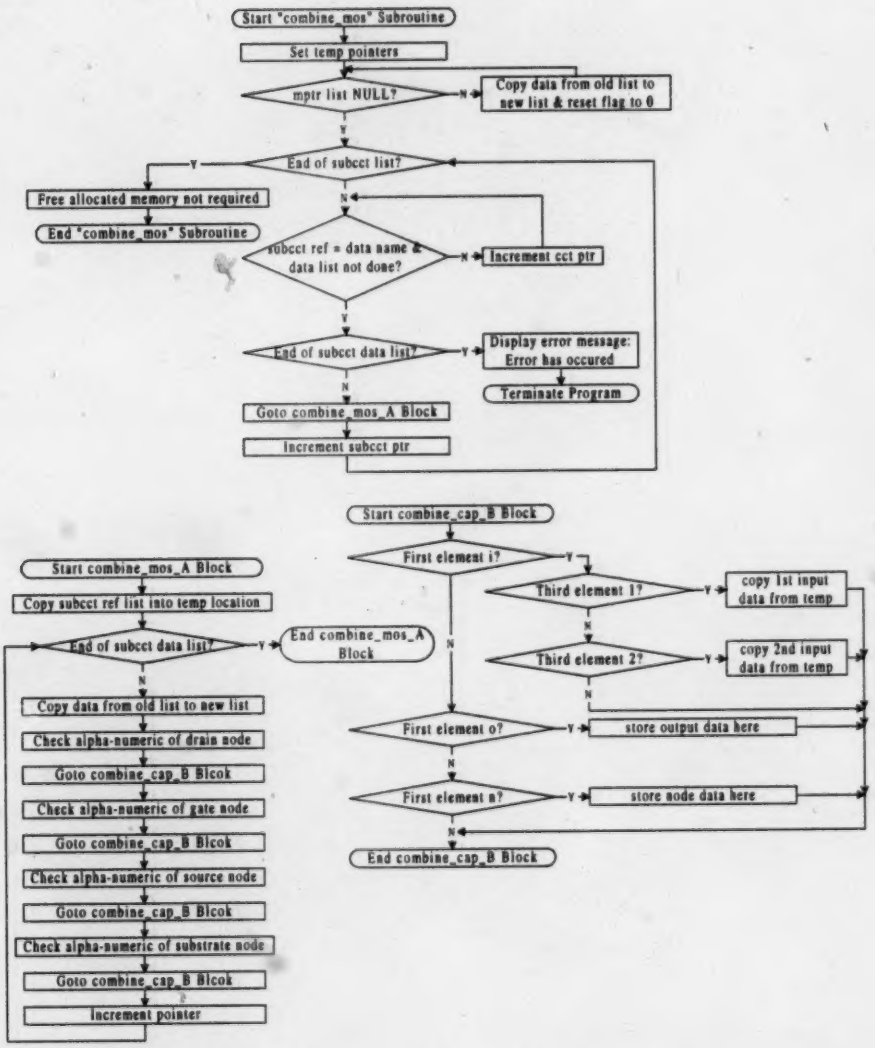

Start "combine_mos" Subroutine
Set temp pointers
mptr list NULL?
Copy data from old list to new list & reset flag to 0
End of subcct list?
Free allocated memory not required
End "combine_mos" Subroutine
subcct ref = data name & data list not done?
Increment cct ptr
End of subcct data list?
Display error message: Error has occured
Terminate Program
Goto combine_mos_A Block
Increment subcct ptr
Start combine_mos_A Block
Copy subcct ref list into temp location
End of subcct data list?
End combine_mos_A Block
Copy data from old list to new list
Check alpha-numeric of drain node
Goto combine_cap_B Blcok
Check alpha-numeric of gate node
Goto combine_cap_B Blcok
Check alpha-numeric of source node
Goto combine_cap_B Blcok
Check alpha-numeric of substrate node
Goto combine_cap_B Blcok
Increment pointer
Start combine_cap_B Block
First element i?
Third element 1?
copy 1st input data from temp
Third element 2?
copy 2nd input data from temp
First element o?
store output data here
First element n?
store node data here
End combine_cap_B Block

Figure C. 18: Flowchart for "combine_mos" subroutine.

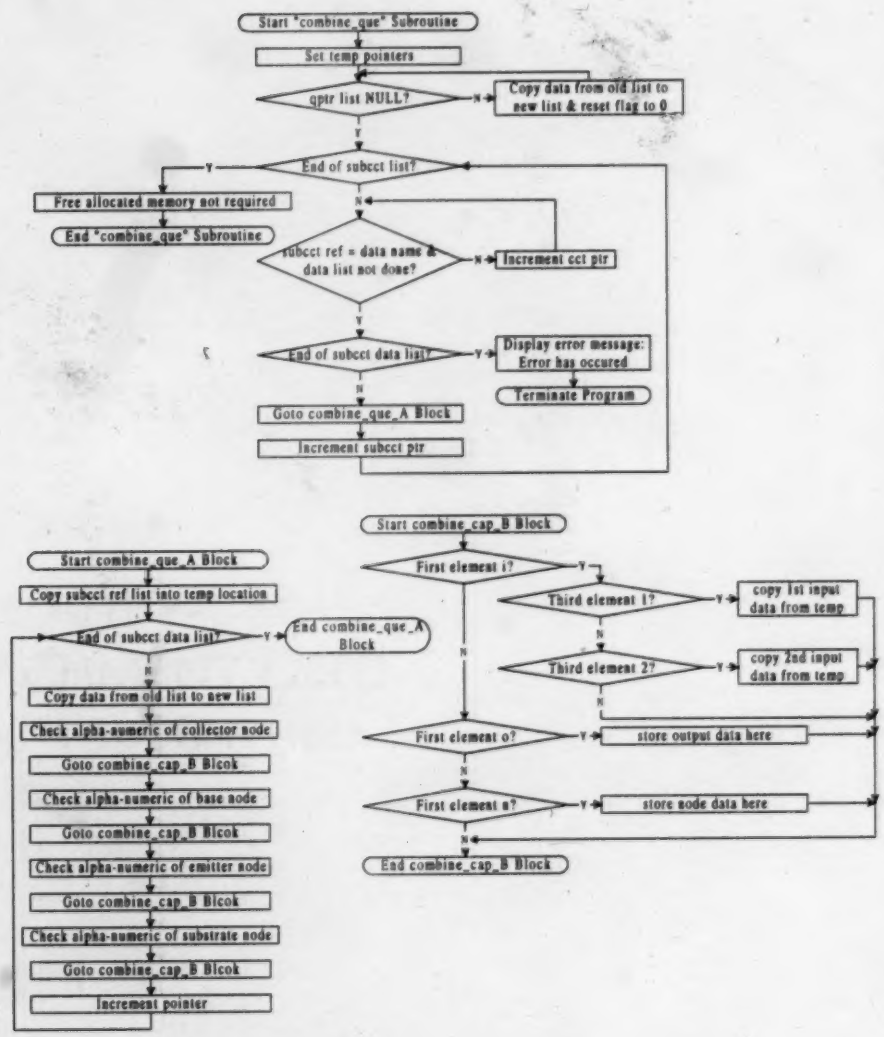

Start "combine_que" Subroutine
Set temp pointers
qptr list NULL?
Copy data from old list to new list & reset flag to 0
End of subcct list?
Free allocated memory not required
End "combine_que" Subroutine
subcct ref = data name & data list not done?
Increment cct ptr
End of subcct data list?
Display error message: Error has occured
Terminate Program
Goto combine_que_A Block
Increment subcct ptr
Start combine_que_A Block
Copy subcct ref list into temp location
End of subcct data list?
End combine_que_A Block
Copy data from old list to new list
Check alpha-numeric of collector node
Goto combine_cap_B Blcok
Check alpha-numeric of base node
Goto combine_cap_B Blcok
Check alpha-numeric of emitter node
Goto combine_cap_B Blcok
Check alpha-numeric of substrate node
Goto combine_cap_B Blcok
Increment pointer
Start combine_cap_B Block
First element i?
Third element 1?
copy 1st input data from temp
Third element 2?
copy 2nd input data from temp
First element o?
store output data here
First element n?
store node data here
End combine_cap_B Block

Figure C.19: Flowchart for "combine_que" subroutine.

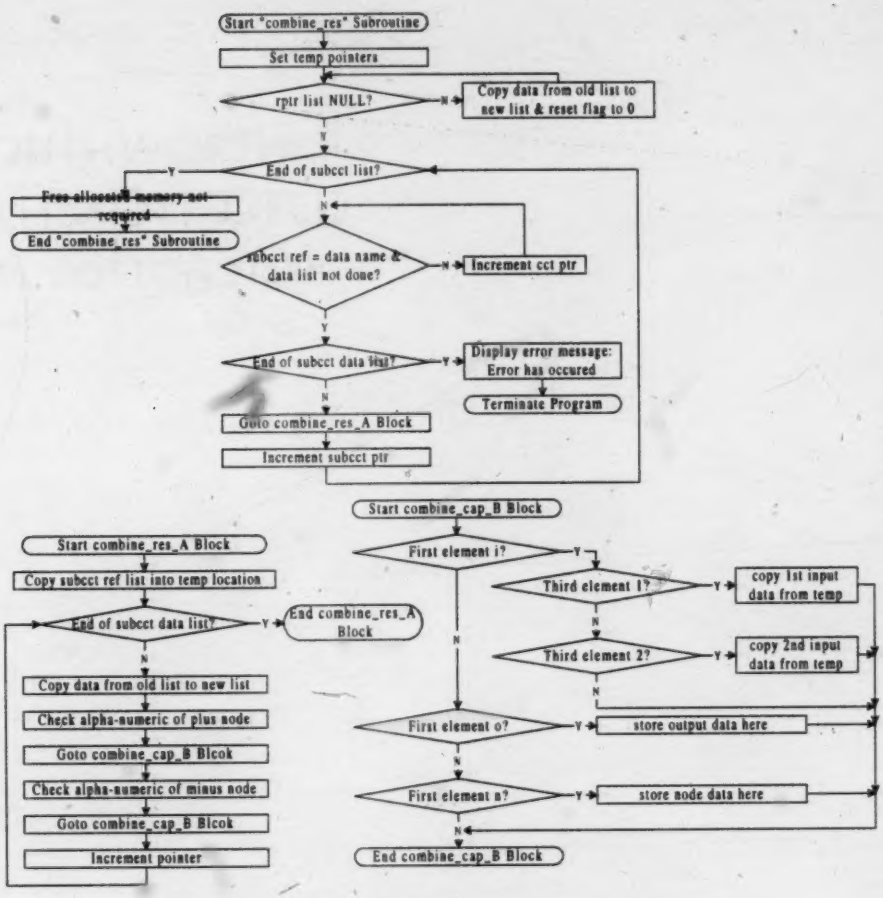

Start "combine_res" Subroutine
Set temp pointers
rptr list NULL?
Copy data from old list to new list & reset flag to 0
End of subcct list?
Free allocated memory not required
End "combine_res" Subroutine
subcct ref = data name & data list not done?
Increment cct ptr
End of subcct data list?
Display error message: Error has occured
Terminate Program
Goto combine_res_A Block
Increment subcct ptr
Start combine_res_A Block
Copy subcct ref list into temp location
End of subcct data list?
End combine_res_A Block
Copy data from old list to new list
Check alpha-numeric of plus node
Goto combine_cap_B Blcok
Check alpha-numeric of minus node
Goto combine_cap_B Blcok
Increment pointer
Start combine_cap_B Block
First element i?
Third element 1?
copy 1st input data from temp
Third element 2?
copy 2nd input data from temp
First element o?
store output data here
First element n?
store node data here
End combine_cap_B Block

Figure C.20: Flowchart for "combine_res" subroutine.

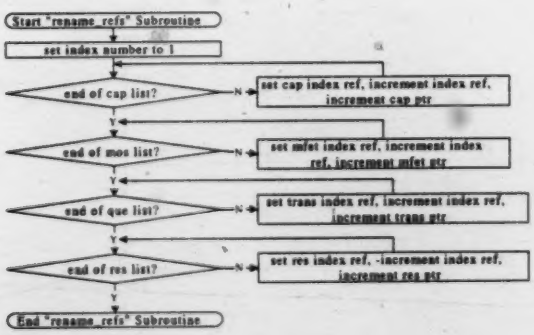

Start "rename_refs" Subroutine
set index number to 1
end of cap list?
set cap index ref, increment index ref, increment cap ptr
end of mos list?
set mfet index ref, increment index ref, increment mfet ptr
end of que list?
set trans index ref, increment index ref, increment trans ptr
end of res list?
set res index ref, -increment index ref, increment res ptr
End "rename_refs" Subroutine

Figure C.21: Flowchart for "rename_refs" subroutine.

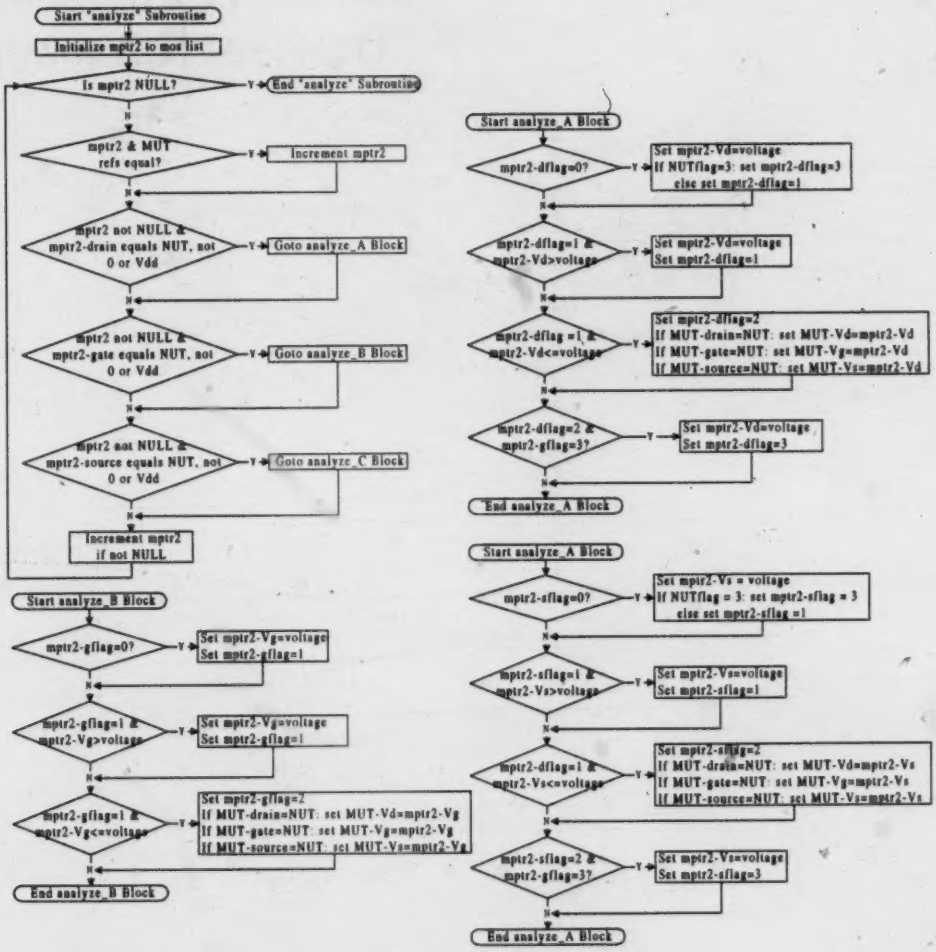

Start "analyze" Subroutine
Initialize mptr2 to mos list
Is mptr2 NULL?
End "analyze" Subroutine
mptr2 & MUT refs equal?
Increment mptr2
mptr2 not NULL & mptr2-drain equals NUT, not 0 or Vdd
Goto analyze_A Block
mptr2 not NULL & mptr2-gate equals NUT, not 0 or Vdd
Goto analyze_B Block
mptr2 not NULL & mptr2-source equals NUT, not 0 or Vdd
Goto analyze_C Block
Increment mptr2 if not NULL
Start analyze_B Block
mptr2-gflag=0?
Set mptr2-Vg=voltage Set mptr2-gflag=1
mptr2-gflag=1 & mptr2-Vg>voltage
Set mptr2-Vg=voltage Set mptr2-gflag=1
mptr2-gflag=1 & mptr2-Vg<voltage
Set mptr2-gflag=2 If MUT-drain=NUT: set MUT-Vd=mptr2-Vg If MUT-gate=NUT: set MUT-Vg=mptr2-Vg If MUT-source=NUT: set MUT-Vs=mptr2-Vg
End analyze_B Block
Start analyze_A Block
mptr2-dflag=0?
Set mptr2-Vd=voltage If NUTflag=3: set mptr2-dflag=3 else set mptr2-dflag=1
mptr2-dflag=1 & mptr2-Vd>voltage
Set mptr2-Vd=voltage Set mptr2-dflag=1
mptr2-dflag=1 & mptr2-Vd<voltage
Set mptr2-dflag=2 If MUT-drain=NUT: set MUT-Vd=mptr2-Vd If MUT-gate=NUT: set MUT-Vg=mptr2-Vd If MUT-source=NUT: set MUT-Vs=mptr2-Vd
mptr2-dflag=2 & mptr2-gflag=3?
Set mptr2-Vd=voltage Set mptr2-dflag=3
End analyze_A Block
Start analyze_A Block
mptr2-sflag=0?
Set mptr2-Vs = voltage If NUTflag = 3: set mptr2-sflag = 3 else set mptr2-sflag =1
mptr2-sflag=1 & mptr2-Vs>voltage
Set mptr2-Vs=voltage Set mptr2-sflag=1
mptr2-dflag=1 & mptr2-Vs<voltage
Set mptr2-sflag=2 If MUT-drain=NUT: set MUT-Vd=mptr2-Vs If MUT-gate=NUT: set MUT-Vg=mptr2-Vs If MUT-source=NUT: set MUT-Vs=mptr2-Vs
mptr2-sflag=2 & mptr2-gflag=3?
Set mptr2-Vs=voltage Set mptr2-sflag=3
End analyze_A Block

Figure C.22: Flowchart for "analyze" subroutine.

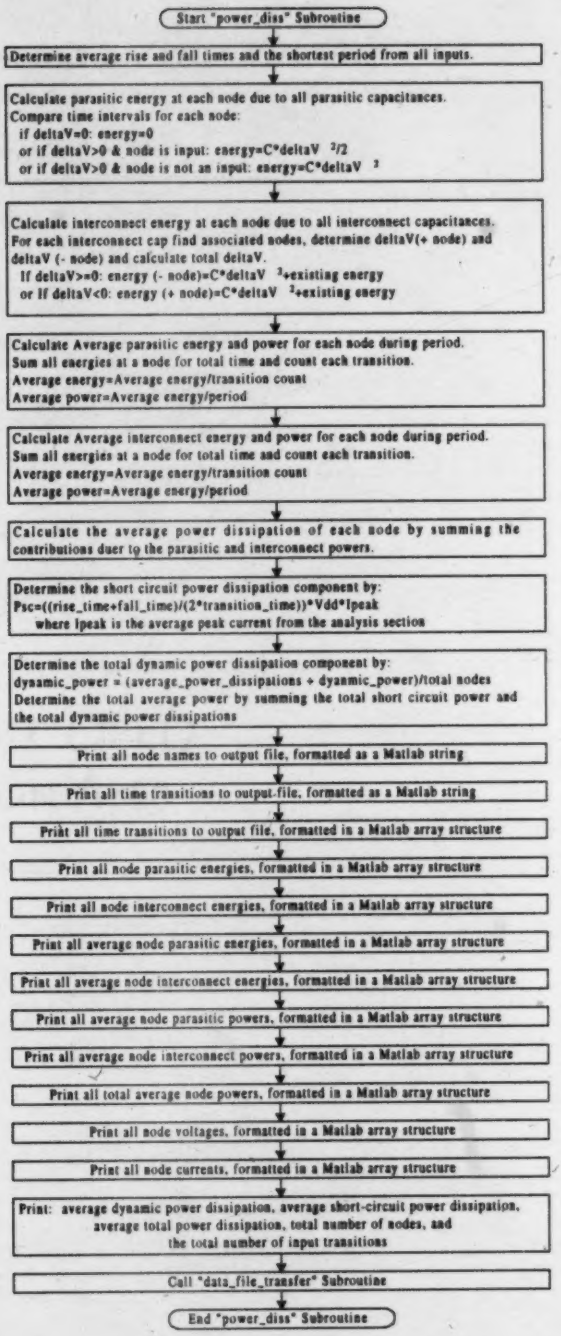

Start "power_diss" Subroutine
Determine average rise and fall times and the shortest period from all inputs.
Calculate parasitic energy at each node due to all parasitic capacitances.
Compare time intervals for each node:
if deltaV=0: energy=0
or if deltaV>0 & node is input: energy=C*deltaV 2/2
or if deltaV>0 & node is not an input: energy=C*deltaV 2
Calculate interconnect energy at each node due to all interconnect capacitances.
For each interconnect cap find associated nodes, determine deltaV(+ node) and deltaV (- node) and calculate total deltaV.
If deltaV>=0: energy (- node)=C*deltaV 2+existing energy
or If deltaV<0: energy (+ node)=C*deltaV 2+existing energy
Calculate Average parasitic energy and power for each node during period.
Sum all energies at a node for total time and count each transition.
Average energy=Average energy/transition count
Average power=Average energy/period
Calculate Average interconnect energy and power for each node during period.
Sum all energies at a node for total time and count each transition.
Average energy=Average energy/transition count
Average power=Average energy/period
Calculate the average power dissipation of each node by summing the contributions duer to the parasitic and interconnect powers.
Determine the short circuit power dissipation component by:
Psc=((rise_time+fall_time)/(2*transition_time))*Vdd*Ipeak
where Ipeak is the average peak current from the analysis section
Determine the total dynamic power dissipation component by:
dynamic_power = (average_power_dissipations + dyanmic_power)/total nodes
Determine the total average power by summing the total short circuit power and the total dynamic power dissipations
Print all node names to output file, formatted as a Matlab string
Print all time transitions to output file, formatted as a Matlab string
Print all time transitions to output file, formatted in a Matlab array structure
Print all node parasitic energies, formatted in a Matlab array structure
Print all node interconnect energies, formatted in a Matlab array structure
Print all average node parasitic energies, formatted in a Matlab array structure
Print all average node interconnect energies, formatted in a Matlab array structure
Print all average node parasitic powers, formatted in a Matlab array structure
Print all average node interconnect powers, formatted in a Matlab array structure
Print all total average node powers, formatted in a Matlab array structure
Print all node voltages, formatted in a Matlab array structure
Print all node currents, formatted in a Matlab array structure
Print: average dynamic power dissipation, average short-circuit power dissipation, average total power dissipation, total number of nodes, and the total number of input transitions
Call "data_file_transfer" Subroutine
End "power_diss" Subroutine

Figure C.23: Flowchart for "power_diss" subroutine.

Appendix C.4 Level – Three and Four Program Flowcharts

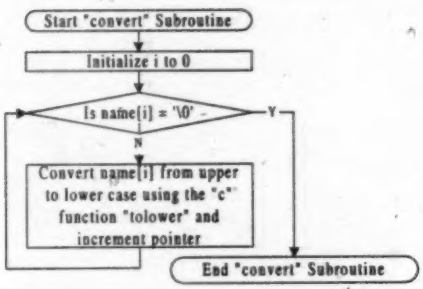

Start "convert" Subroutine
Initialize i to 0
Is name[i] = '\0' -
Y
N
Convert name[i] from upper to lower case using the "c" function "tolower" and increment pointer
End "convert" Subroutine

Figure C.24: Flowchart for "convert" subroutine.

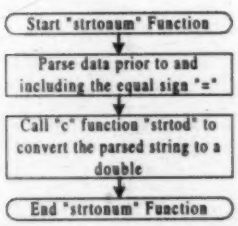

Start "strtonum" Function
Parse data prior to and including the equal sign "="
Call "c" function "strtod" to convert the parsed string to a double
End "strtonum" Function

Figure C.25: Flowchart for "strtonum" function.

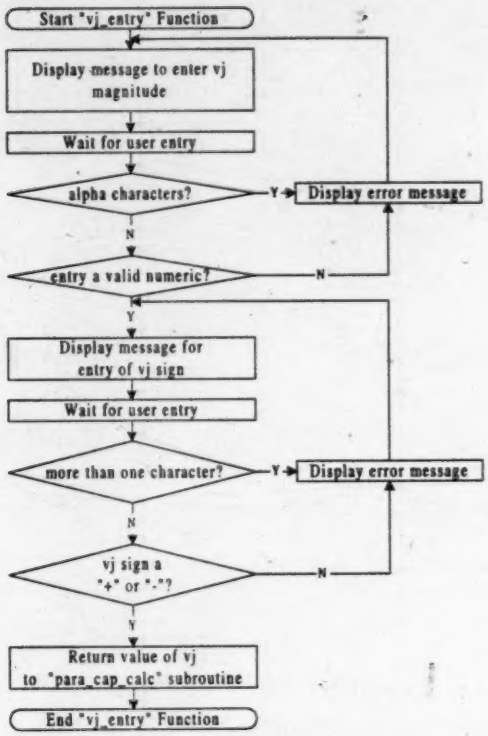


Figure C.26: Flowchart for "vj_entry" function.

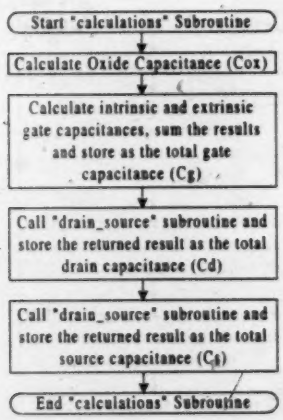

Start "calculations" Subroutine
Calculate Oxide Capacitance (Cox)
Calculate intrinsic and extrinsic gate capacitances, sum the results and store as the total gate capacitance (Cg)
Call "drain_source" subroutine and store the returned result as the total drain capacitance (Cd)
Call "drain_source" subroutine and store the returned result as the total source capacitance (Cs)
End "calculations" Subroutine

Figure C.27: Flowchart for "calculations" subroutine.

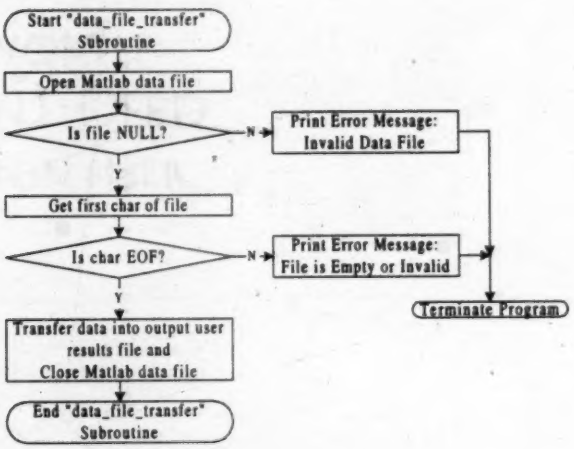


Figure C.28: Flowchart for "data_file_transfer" subroutine.

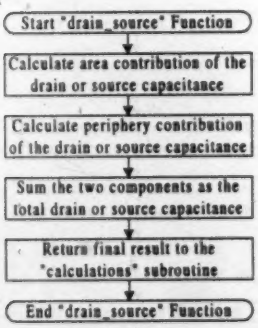

Start "drain_source" Function
Calculate area contribution of the drain or source capacitance
Calculate periphery contribution of the drain or source capacitance
Sum the two components as the total drain or source capacitance
Return final result to the "calculations" subroutine
End "drain_source" Function

Figure C.29: Flowchart for "drain_source" function.

Appendix C.5 Matlab Program Source Code

```
%%%
```

```
  elseif start_node == end_node
    for col = 1:max_times
      voltage(newrow,col) = node_voltage(start_node,col);
      para_E(newrow,col) = node_para_energy(start_node,col);
      inter_E(newrow,col) = node_inter_energy(start_node,col);
    end
    ave_para_E = ave_para_energy(start_node);
    ave_inter_E = ave_inter_energy(start_node);
    ave_para_P = ave_node_para_power(start_node);
    ave_inter_P = ave_node_inter_power(start_node);
    ave_dyn_P = ave_node_power(start_node);
  end
elseif range_type == 2
  start_node = 1;
  end_node = max_nodes;
  plot_loop = 1;
  voltage = node_voltage';
  para_E = node_para_energy';
  inter_E = node_inter_energy';
  ave_para_E = ave_para_energy';
  ave_inter_E = ave_inter_energy';
  ave_para_P = ave_node_para_power';
  ave_inter_P = ave_node_inter_power';
  ave_dyn_P = ave_node_power';
  node_label = node_axis_label;
elseif range_type == 3
  button = questdlg('Are you sure you want to quit?','Exit Program','Yes','No','No');
  if strcmp(button,'Yes')
    program_loop = 2;
    plot_loop == 2;
    clc;
  end
end
%%%
```

```
elseif plot_type == 2
  if plotted(2) == 1
    figure(2);
    clf;
    rotate3d off;
  else
    plotted(2) = 1;
  end
  figure(plot_type);
  axes('Position',[0.15,0.15,0.75,0.75]);
  bar3(para_E,0.5,'detached');
  set(gca,'Xtick',[1:xtick_length]);
  set(gca,'XtickLabel',{node_label});
  set(gca,'Xlabel',text('String','Node'));
  set(gca,'Ytick',[1:max_times]);
  set(gca,'YtickLabel',{time_axis_label});
  set(gca,'Ylabel',text('String','Time'));
  set(gca,'Zlabel',text('String',' Node Parasitic Energy'));
  set(gca,'FontName','TimesNewRoman','FontSize',fnt_size);
  if xtick_length > 1
    set(gca,'XLim',[0 xtick_length+1],'YLim',[0 max_times+1]);
  end
  set(figure(plot_type),'MenuBar','none');
  figrote2 = uicontrol('Style','Radio','String','rotate','Position',[1 31 60 20],'Callback','rot82');
  clrclse2 = uicontrol('Style','Popup','String','Press|Print|Close','Position',[1 10 60 20],'Callback','close2');

elseif plot_type == 3
  if plotted(3) == 1
    figure(3);
    clf;
    rotate3d off;
  else
    plotted(3) = 1;
  end
  figure(plot_type);
  axes('Position',[0.15,0.15,0.75,0.75]);
  bar3(inter_E,0.5,'detached');
  set(gca,'Xtick',[1:xtick_length]);
  set(gca,'XtickLabel',{node_label});
  set(gca,'Xlabel',text('String','Node'));
  set(gca,'Ytick',[1:max_times]);
  set(gca,'YtickLabel',{time_axis_label});
  set(gca,'Ylabel',text('String','Time'));
  set(gca,'Zlabel',text('String','Node Interconnect Energy'));
  set(gca,'FontName','TimesNewRoman','FontSize',fnt_size);
  if xtick_length > 1
    set(gca,'XLim',[0 xtick_length+1],'YLim',[0 max_times+1]);
  end
  set(figure(plot_type),'MenuBar','none');
  figrote3 = uicontrol('Style','Radio','String','rotate','Position',[1 31 60 20],'Callback','rot83');
  clrclse3 = uicontrol('Style','Popup','String','Press|Print|Close','Position',[1 10 60 20],'Callback','close3');

elseif plot_type == 4
  if plotted(4) == 1
    figure(4);
    clf;
    rotate3d off;
  else
    plotted(4) = 1;
  end
  figure(plot_type);
  axes('Position',[0.15,0.15,0.75,0.75]);
  bar3(ave_para_E,0.5,'detached');
  time_name_length = size(time_axis_label);
  time_label = time_axis_label(max_times,1:time_name_length(2));
  set(gca,'Xtick',[1]);
  set(gca,'XtickLabel',{time_label});
  set(gca,'Xlabel',text('String','Period'));
```

```
    set(gca,'Ytick',[1:xtick_length]);
    set(gca,'YtickLabel',{node_label});
    set(gca,'Ylabel',text('String','Node'));
    set(gca,'Zlabel',text('String','Node Average Parasitic Energy'));
    set(gca,'FontName','TimesNewRoman','FontSize',fnt_size);
    set(figure(plot_type),'MenuBar','none');
    figrote4 = uicontrol('Style','Radio','String','rotate','Position',[1 31 60 20],'Callback','rot84');
    clrclse4 = uicontrol('Style','Popup','String','Press|Print|Close','Position',[1 10 60 20],'Callback','close4');

elseif plot_type == 5
    if plotted(5) == 1
       figure(5);
       clf;
       rotate3d off;
    else
       plotted(5) = 1;
    end
    figure(plot_type);
    axes('Position',[0.15,0.15,0.75,0.75]);
    bar3(ave_inter_E,0.5,'detached');
    time_name_length = size(time_axis_label);
    time_label = time_axis_label(max_times,1:time_name_length(2));
    set(gca,'Xtick',[1]);
    set(gca,'XtickLabel',{time_label});
    set(gca,'Xlabel',text('String','Period'));
    set(gca,'Ytick',[1:xtick_length]);
    set(gca,'YtickLabel',{node_label});
    set(gca,'Ylabel',text('String','Node'));
    set(gca,'Zlabel',text('String','Node Average Interconnect Energy'));
    set(gca,'FontName','TimesNewRoman','FontSize',fnt_size);
    set(figure(plot_type),'MenuBar','none');
    figrote5 = uicontrol('Style','Radio','String','rotate','Position',[1 31 60 20],'Callback','rot85');
    clrclse5 = uicontrol('Style','Popup','String','Press|Print|Close','Position',[1 10 60 20],'Callback','close5');

elseif plot_type == 6
    if plotted(6) == 1
       figure(6);
       clf;
       rotate3d off;
    else
       plotted(6) = 1;
    end
    figure(plot_type);
    axes('Position',[0.15,0.15,0.75,0.75]);
    bar3(ave_para_P,0.5,'detached');
    time_name_length = size(time_axis_label);
    time_label = time_axis_label(max_times,1:time_name_length(2));
    set(gca,'Xtick',[1]);
    set(gca,'XtickLabel',{time_label});
    set(gca,'Xlabel',text('String','Period'));
    set(gca,'Ytick',[1:xtick_length]);
    set(gca,'YtickLabel',{node_label});
    set(gca,'Ylabel',text('String','Node'));
    set(gca,'Zlabel',text('String','Node Average Parasitic Power'));
    set(gca,'FontName','TimesNewRoman','FontSize',fnt_size);
    set(figure(plot_type),'MenuBar','none');
    figrote6 = uicontrol('Style','Radio','String','rotate','Position',[1 31 60 20],'Callback','rot86');
    clrclse6 = uicontrol('Style','Popup','String','Press|Print|Close','Position',[1 10 60 20],'Callback','close6');

elseif plot_type == 7
    if plotted(7) == 1
       figure(7);
       clf;
       rotate3d off;
    else
       plotted(7) = 1;
    end
    figure(plot_type);
```

```
    axes('Position',[0.15,0.15,0.75,0.75]);
    bar3(ave_inter_P,0.5,'detached');
    time_name_length = size(time_axis_label);
    time_label = time_axis_label(max_times,1:time_name_length(2));
    set(gca,'Xtick',[1]);
    set(gca,'XtickLabel',{time_label});
    set(gca,'Xlabel',text('String','Period'));
    set(gca,'Ytick',[1:xtick_length]);
    set(gca,'YtickLabel',{node_label});
    set(gca,'Ylabel',text('String','Node'));
    set(gca,'Zlabel',text('String','Node Average Interconnect Power'));
    set(gca,'FontName','TimesNewRoman','FontSize',fnt_size);
    set(figure(plot_type),'MenuBar','none');
    figrote7 = uicontrol('Style','Radio','String','rotate','Position',[1 31 60 20],'Callback','rot87');
    clrclse7 = uicontrol('Style','Popup','String','Press|Print|Close','Position',[1 10 60 20],'Callback','close7');

  elseif plot_type == 8
    if plotted(8) == 1
      figure(8);
      clf;
      rotate3d off;
    else
      plotted(8) = 1;
    end
    figure(plot_type);
    axes('Position',[0.15,0.15,0.75,0.75]);
    bar3(ave_dyn_P,0.5,'detached');
    time_name_length = size(time_axis_label);
    time_label = time_axis_label(max_times,1:time_name_length(2));
    set(gca,'Xtick',[1]);
    set(gca,'XtickLabel',{time_label});
    set(gca,'Xlabel',text('String','Period'));
    set(gca,'Ytick',[1:xtick_length]);
    set(gca,'YtickLabel',{node_label});
    set(gca,'Ylabel',text('String','Node'));
    set(gca,'Zlabel',text('String','Average Node Dynamic Power'));
    set(gca,'FontName','TimesNewRoman','FontSize',fnt_size);
    set(figure(plot_type),'MenuBar','none');
    figrote8 = uicontrol('Style','Radio','String','rotate','Position',[1 31 60 20],'Callback','rot88');
    clrclse8 = uicontrol('Style','Popup','String','Press|Print|Close','Position',[1 10 60 20],'Callback','close8');

  elseif plot_type == 9
    plot_loop = 2;
  end
 end
 for i = 1:8
  if plotted(i) == 1
    close(i);
  end
 end
end
```

Appendix D.1 T-Cell XNOR Netlist File

The following netlist is for a T-Cell Exclusive NOR gate.

```
* # FILE NAME: /SIMULATION/TESTING_CIRCUIT/SPECTRES/
xi31 a2 a1 output vdd vss tcells_txnr2_extracted
vi3 a2 vss pulse 0.0 5.0 1.0e-6 1e-9 1e-9 1e-6 2e-6
vi2 a1 vss pulse 0.0 5.0 500e-9 1e-9 1e-9 1e-6 2e-6
.SUBCKT tcells_txnr2_extracted ip1 ip2 op vdd vss
c8 ip1 vdd 2.603040031855e-15
c10 n5 vdd 1.27776000060319e-15
c12 ip2 vdd 2.603040031855e-15
c14 n2 vdd 2.603040031855e-15
c16 ip1 0 2.65647997577199e-15
c18 n5 0 1.33120005039929e-15
c20 ip2 0 2.65647997577199e-15
c22 n2 0 2.65647997577199e-15
c24 vss 0 5.95903983777635e-15
c26 ip1 n5 170.815999335532e-18
c28 ip1 n3 170.815999335532e-18
c30 ip1 n2 170.815999335532e-18
c32 ip1 vss 509.919975235956e-18
c34 ip1 vdd 339.104002370204e-18
c36 n5 vss 633.510976086305e-18
c38 n5 vdd 462.694976750773e-18
c40 ip2 n3 170.815999335532e-18
c42 ip2 n2 170.815999335532e-18
c44 ip2 vss 725.039979938065e-18
c46 ip2 vdd 339.104002370204e-18
c48 op vss 341.631998671064e-18
c50 op vdd 341.631998671064e-18
c52 n2 vss 339.104002370204e-18
c54 n2 vdd 509.919975235956e-18
c56 n3 op 220.627995260308e-18
c58 n3 n2 870.900006367322e-18
c60 op n2 267.075999835063e-18
c62 op vdd 174.180003920442e-18
c64 n2 vdd 266.649995202153e-18
c66 ip1 n1 549.538406096559e-18
c68 n1 vss 419.959994039866e-18
c70 n5 vss 240.736001562753e-18
c72 ip2 vss 309.273381322259e-18
c74 op vss 348.360007840885e-18
c76 n2 vss 325.136005553507e-18
c78 ip1 vdd 185.832000901109e-18
c80 n5 vdd 274.40699058421e-18
c82 ip2 vdd 185.832000901109e-18
c84 op vdd 1.08401995452273e-15
c86 n2 vdd 615.243016512871e-18
c88 ip1 0 150.662994407764e-18
c90 n5 0 260.223000365963e-18
c92 ip2 0 292.401993790391e-18
c94 op 0 1.06337204412575e-15
c96 n2 0 361.339993664169e-18
c98 vss 0 11.0279437627115e-15
c100 vss vdd 2.53954390102998e-15
c102 vss vdd 686.088005188274e-18
c104 ip1 n1 47.6519990435507e-18
c106 ip2 vss 47.6519990435507e-18
c108 vss vdd 243.832012864086e-18
c110 vss 0 294.279998183352e-18
m112 n6 ip1 n5 vdd mpch_0p8 L=800.000009348878e-9 W=3.00000010611257e-6
+AD=4.92000017116023e-12 AS=8.33999969779287e-12 PD=6.59999977870029e-6
+PS=12.2000001283595e-6
```

```
m114 vdd ip2 n6 vdd mpch_0p8 L=800.000009348878e-9 W=3.00000010611257e-6
+AD=8.33999969779287e-12 AS=4.92000017116023e-12 PD=12.2000001283595e-6
+PS=6.59999977870029e-6
m116 vdd n2 op vdd mpch_0p8 L=800.000009348878e-9 W=3.00000010611257e-6
+AD=8.33999969779287e-12 AS=8.33999969779287e-12 PD=12.2000001283595e-6
+PS=12.2000001283595e-6
m118 n3 n5 vdd vdd mpch_0p8 L=800.000009348878e-9 W=11.7999998110463e-6
+AD=16.5200006452226e-12 AS=27.1400003165612e-12 PD=14.6000002132496e-6
+PS=28.2000000879634e-6
m120 n2 ip1 n3 vdd mpch_0p8 L=800.000009348878e-9 W=11.7999998110463e-6
+AD=16.5200006452226e-12 AS=16.5200006452226e-12 PD=14.6000002132496e-6
+PS=14.6000002132496e-6
m122 n3 ip2 n2 vdd mpch_0p8 L=800.000009348878e-9 W=11.7999998110463e-6
+AD=27.1400003165612e-12 AS=16.5200006452226e-12 PD=28.2000000879634e-6
+PS=14.6000002132496e-6
m124 vdd n2 op vdd mpch_0p8 L=800.000009348878e-9 W=11.7999998110463e-6
+AD=27.1400003165612e-12 AS=27.1400003165612e-12 PD=28.2000000879634e-6
+PS=28.2000000879634e-6
m126 n2 n5 vss 0 mnch_0p8 L=800.000009348878e-9 W=8.79999970493373e-6
+AD=12.3200000548551e-12 AS=20.2400006477088e-12 PD=11.600000107137e-6
+PS=22.1999998757383e-6
m128 n1 ip1 n2 0 mnch_0p8 L=800.000009348878e-9 W=8.79999970493373e-6
+AD=12.3200000548551e-12 AS=12.3200000548551e-12 PD=11.600000107137e-6
+PS=11.600000107137e-6
m130 vss ip2 n1 0 mnch_0p8 L=800.000009348878e-9 W=8.79999970493373e-6
+AD=20.2400006477088e-12 AS=12.3200000548551e-12 PD=22.1999998757383e-6
+PS=11.600000107137e-6
m132 vss n2 op 0 mnch_0p8 L=800.000009348878e-9 W=8.79999970493373e-6
+AD=20.2400006477088e-12 AS=20.2400006477088e-12 PD=22.1999998757383e-6
+PS=22.1999998757383e-6
m134 n5 ip1 vss 0 mnch_0p8 L=800.000009348878e-9 W=3.00000010611257e-6
+AD=4.92000017116023e-12 AS=8.33999969779287e-12 PD=6.59999977870029e-6
+PS=12.2000001283595e-6
m136 vss ip2 n5 0 mnch_0p8 L=800.000009348878e-9 W=3.00000010611257e-6
+AD=8.33999969779287e-12 AS=4.92000017116023e-12 PD=12.2000001283595e-6
+PS=6.59999977870029e-6
m138 vss n2 op 0 mnch_0p8 L=800.000009348878e-9 W=3.00000010611257e-6
+AD=8.33999969779287e-12 AS=8.33999969779287e-12 PD=12.2000001283595e-6
+PS=12.2000001283595e-6
r140 vss 0 1.0
r142 vss 0 1.0
.ENDS tcells_txnr2_extracted
.end
```

Appendix D.2 T-Cell XNOR Schematic Diagram

The following schematic is a representation of the test circuit from Chapter 5. It is a T-Cell XNOR gate, which is a standard cell layout from Cadence.

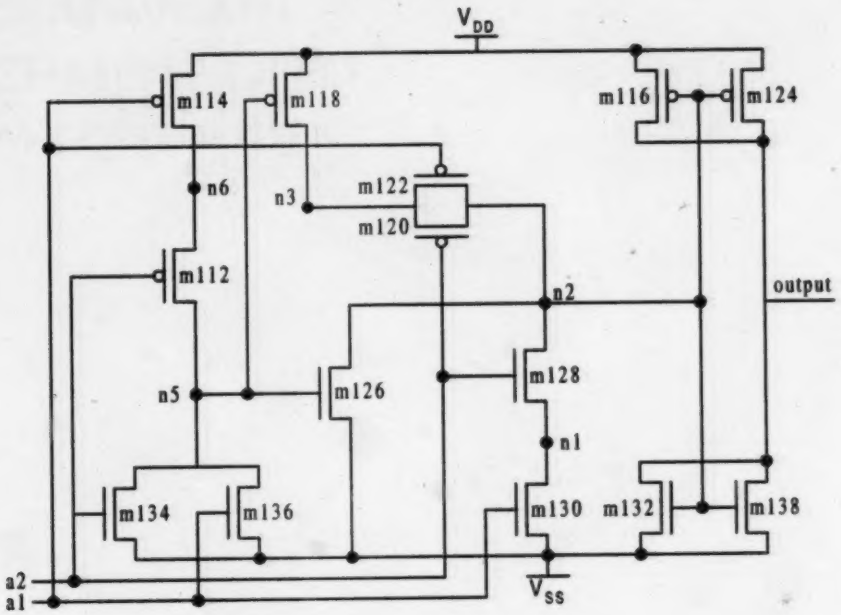


Figure D.1: T-Cell XNOR schematic diagram.

Appendix D.3 K-Cell Compressor Netlist File

The following netlist is for a K-Cell Based Compressor Circuit. It consists of the following gate subcircuit structures: XOR, OR, NOR, NAND.

```
* # FILE NAME: /SIMULATION/KCOMP/SPECTRES/SCHEMATIC/
xi75 net44 cin sum vdd vss kcells_kxor2_1_b_extracted
xi76 net99 net84 net44 vdd vss kcells_kxor2_1_b_extracted
xi73 d1 d2 net49 vdd vss kcells_kor2_1_extracted
xi74 d3 d4 net54 vdd vss kcells_kor2_1_extracted
xi71 net99 net84 net59 vdd vss kcells_knr2_1_extracted
xi72 net74 net89 net64 vdd vss kcells_knr2_1_extracted
xi77 net64 net59 net69 vdd vss kcells_knr2_1_extracted
xi64 d3 d4 net74 vdd vss kcells_knd2_1_extracted
xi65 net69 net104 carry vdd vss kcells_knd2_1_extracted
xi66 net74 net54 net84 vdd vss kcells_knd2_1_extracted
xi67 d1 d2 net89 vdd vss kcells_knd2_1_extracted
xi68 net89 net74 cout vdd vss kcells_knd2_1_extracted
xi69 net89 net49 net99 vdd vss kcells_knd2_1_extracted
xi70 net44 cin net104 vdd vss kcells_knd2_1_extracted
vi49 d3  vss pulse 0.0 5.0 640e-9 1e-9 1e-9 860e-9 2e-6
vi60 d4  vss pulse 0.0 5.0 800e-9 1e-9 1e-9 320e-9 2e-6
vi61 cin vss pulse 0.0 5.0 480e-9 1e-9 1e-9 1.2e-6 2e-6
vi3  d2  vss pulse 0.0 5.0 160e-9 1e-9 1e-9 1.84e-6 2e-6
vi2  d1  vss pulse 0.0 5.0 320e-9 1e-9 1e-9 1.52e-6 2e-6
.SUBCKT kcells_kxor2_1_b_extracted ip1 ip2 op vdd vss
c7 vss 0 5.116679971292O5e-15
c9 op 0 688.344024503727e-18
c11 n5 0 199.72799991729e-18
c13 n3 0 753.030024861406e-18
c15 n2 0 437.375997287039e-18
c17 n1 0 216.803998845479e-18
c19 ip1 0 87.368000820488e-18
c21 op vdd 249.22800978479e-18
c23 n5 vdd 859.887995717864e-18
c25 ip2 vdd 141.448003860576e-18
c27 n3 vdd 573.756983882691e-18
c29 n2 vdd 2.56287605883681e-15
c31 n1 vdd 464.579995635328e-18
c33 ip1 vdd 54.0799997313655e-18
c35 n5 vdd 241.567999675194e-18
c37 n3 5 217.983995150343e-18
c39 n3 ip2 206.191999505363e-18
c41 n2 vdd 241.567999675194e-18
c43 n2 n5 170.815999335532e-18
c45 n2 ip2 170.815999335532e-18
c47 n2 n3 206.191999505363e-18
c49 ip1 n2 170.815999335532e-18
c51 n5 0 903.440047585516e-18
c53 ip2 0 2.74711991266649e-15
c55 n3 0 938.835013352432e-18
c57 n2 0 409.12001048804e-18
c59 ip1 0 558.990023238497e-18
c61 n5 vdd 2.717119911498870-15
c63 ip2 vdd 2.47664005799332e-15
c65 n3 vdd 1.23884497766572e-15
c67 n2 vdd 2.16192009600154e-15
c69 ip1 vdd 724.450021490303e-18
m71 op n5 vdd vdd mpch_0p8 L=800.000009348878e-9 W=6.10000006417977e-6
+AD=12.8099995888586e-12 AS=7.45999963192023e-12 PD=16.3999993674224e-6
+PS=8.79999970493373e-6
m73 vdd ip2 n3 vdd mpch_0p8 L=800.000009348878e-9 W=3.00000010611257e-6
+AD=7.45999963192023e-12 AS=5.58000000372427e-12 PD=8.79999970493373e-6
```

```
+PS=10.2000003607827e-6
m75 n1 n2 vdd vdd mpch_0p8 L=800.000009348878e-9 W=6.1999999161344e-6
+AD=6.44999982243877e-12 AS=7.4400000049657e-12 PD=9.200000022246976-6
+PS=8.600000000102445e-6
m77 vdd ip1 n2 vdd mpch_0p8 L=800.000009348878e-9 W=6.1999999161344e-6
+AD=7.4400000049657e-12 AS=13.02000000869e-12 PD=8.600000000102445e-6
+PS=16.6000008903211e-6
m79 n2 ip2 n5 vdd mpch_0p8 L=800.000009348878e-9 W=2.799999994745594e-6
+AD=5.27999996155493e-12 AS=3.35999999524772e-12 PD=9.800000043469486-6
+PS=5.2000000032346e-6
m81 n5 n3 n1 vdd mpch_0p8 L=800.000009348878e-9 W=2.799999994745594e-6
+AD=3.359999995247726-12 AS=6.44999982243877e-12 PD=5.200000032346e-6
+PS=9.200000022246976-6
m83 op n5 vss 0 mnch_0p8 L=800.000009348878e-9 W=4.500000159168856-6
+AD=9.44999981045136e-12 AS=5.500000194863526-12 PD=13.1999995574006e-6
+PS=7.199999799922816-6
m85 vss ip2 n3 0 mnch_0p8 L=800.000009348878e-9 W=2.199999926233436-6
+AD=5.500000194863526-12 AS=4.620000128990896-12 PD=7.199999799922816-6
+PS=8.600000000102445e-6
m87 n1 n2 vss 0 mnch_0p8 L=800.000009348878e-9 W=3.800000058618026-6
+AD=4.740000059122456-12 AS=4.559999947084676-12 PD=6.400000074791026-6
+PS=6.1999999161344e-6
m89 vss ip1 n2 0 mnch_0p8 L=800.000009348878e-9 W=3.800000058618026-6
+AD=4.559999947084676-12 AS=7.980000341079046-12 PD=6.1999999161344e-6
+PS=11.7999998110463e-6
m91 n2 n3 n5 0 mnch_0p8  L=800.000009348878e-9 W=3.000000106112576-6
+AD=6.300000018194536-12 AS=3.600000072351286-12 PD=10.2000003607827e-6
+PS=5.400000191002626-6
m93 n5 ip2 n1 0 mnch_0p8 L=800.000009348878e-9 W=3.000000106112576-6
+AD=3.600000072351286-12 AS=4.740000059122456-12 PD=5.400000191002626-6
+PS=6.400000074791026-6
r95 vss 0 1.0
.ENDS kcells_kxor2_1_b_extracted
.SUBCKT kcells_knd2_1_extracted ip1 ip2 op vdd vss
c3 vss 0 2.28988602202446e-15
c5 op 0 323.468991773751e-18
c7 ip2 0 145.985003493646e-18
c9 op vdd 441.57000151603e-18
c11 ip1 vdd 141.448003860576e-18
c13 ip2 vdd 102.274999833665e-18
c15 ip1 op 456.146988552024e-18
c17 ip1 0 452.650013029284e-18
c19 ip2 0 494.759991304236e-18
c21 ip1 vdd 1.18505002106472e-15
c23 ip2 vdd 734.359989335823e-18
m25 vdd ip1 op vdd mpch_0p8 L=800.000009348878e-9 W=5.299999884300636-6
+AD=11.129999699655e-12 AS=6.360000200100756-12  PD=14.7999999171589e-6
+PS=7.700000423938046-6
m27 op ip2 vdd vdd mpch_0p8 L=800.000009348878e-9 W=5.299999884300636-6
+AD=6.360000200100756-12 AS=11.129999699655e-12  PD=7.700000423938046-6
+PS=14.7999999171589e-6
m29 op ip1 n1 0 mnch_0p8 L=800.000009348878e-9 W=5.299999884300636-6
+AD=11.129999699655e-12 AS=3.975000016642756-12 PD=14.7999999171589e-6
+PS=6.799999937356926-6
m31 n1 ip2 vss 0 mnch_0p8 L=800.000009348878e-9 W=5.299999884300636-6
+AD=3.975000016642756-12 AS=11.129999699655e-12 PD=6.799999937356926-6
+PS=14.7999999171589e-6
r33 vss 0 1.0
.ENDS kcells_knd2_1_extracted
.SUBCKT kcells_knr2_1_extracted ip1 ip2 op vdd vss
c3 vss 0 3.043647936291856-15
c5 op 0 390.819996011509e-18
c7 ip1 vdd 319.468005177616e-18
c9 op vdd 126.400000331466-18
c11 ip2 vdd 59.3399982331984e-18
c13 n1 op 166.020006143956-18
c15 ip2 vdd 254.611989425555e-18
c17 op ip1 508.468001945685e-18
```

```
c19 ip1 0 464.064999603398e-18
c21 ip2 0 508.219980110817e-18
c23 ip1 vdd 898.655052587358e-18
c25 ip2 vdd 655.859993339169e-18
m27 op ip1 n1 vdd mpch_0p8 L=800.000009348878e-9 W=6.799999937356926-6
+AD=14.2799999255927e-12 AS=5.09999984951715e-12 PD=17.7999991137767e-6
+PS=8.30000044516055e-6
m29 n1 ip2 vdd vdd mpch_0p8 L=800.000009348878e-9 W=6.799999937356926-6
+AD=5.09999984951715e-12 AS=14.2799999255927e-12 PD=8.30000044516055e-6
+PS=17.7999991137767e-6
m31 vss ip1 op 0 mnch_0p8 L=800.000009348878e-9 W=3.800000058618026-6
+AD=7.980000341079046-12 AS=4.55999994708467e-12 PD=11.7999998110463e-6
+PS=6.1999999161344e-6
m33 op ip2 vss 0 mnch_0p8 L=800.000009348878e-9 W=3.800000058618026-6
+AD=4.55999994708467e-12 AS=7.980000341079046-12 PD=6.1999999161344e-6
+PS=11.7999998110463e-6
r35 vss 0 1.0
.ENDS kcells_knr2_1_extracted
.SUBCKT kcells_kor2_1_extracted ip1 ip2 op vdd vss
c4 vss 0 2.91630394421045e-15
c6 op 0 505.875975247968e-18
c8 n2 0 505.875975247968e-18
c10 ip2 0 141.448003860576e-18
c12 op vdd 580.748023252414e-18
c14 n2 vdd 653.892018165342e-18
c16 ip1 vdd 212.655999093436e-18
c18 n1 n2 139.344000489376e-18
c20 ip2 n2 265.151990965154e-18
c22 ip1 n2 170.815999335532e-18
c24 n2 0 783.200014953624e-18
c26 ip2 0 1.034640051280176-15
c28 ip1 0 783.200014953624e-18
c30 n2 vdd 886.799979457587e-18
c32 ip2 vdd 743.120004076327e-18
c34 ip1 vdd 1.048440017935956-15
m36 op n2 vdd vdd mpch_0p8 L=800.000009348878e-9 W=6.799999937356926-6
+AD=14.2799999255927e-12 AS=8.15999958575508e-12 PD=17.7999991137767e-6
+PS=9.20000002224697e-6
m38 vdd ip2 n1 vdd mpch_0p8 L=800.000009348878e-9 W=6.799999937356926-6
+AD=8.15999958575508e-12 AS=5.09999984951715e-12 PD=9.20000002224697e-6
+PS=8.30000044516055e-6
m40 n1 ip1 n2 vdd mpch_0p8  L=800.000009348878e-9 W=6.799999937356926-6
+AD=5.09999984951715e-12 AS=14.2799999255927e-12 PD=8.30000044516055e-6
+PS=17.7999991137767e-6
m42 op n2 vss 0 mnch_0p8 L=800.000009348878e-9 W=3.800000058618026-6
+AD=7.980000341079046-12 AS=4.55999994708467e-12 PD=11.7999998110463e-6
+PS=6.1999999161344e-6
m44 vss ip2 n2 0 mnch_0p8 L=800.000009348878e-9 W=3.800000058618026-6
+AD=4.55999994708467e-12 AS=4.55999994708467e-12 PD=6.1999999161344e-6
+PS=6.1999999161344e-6
m46 n2 ip1 vss 0 mnch_0p8 L=800.000009348878e-9 W=3.800000058618026-6
+AD=4.55999994708467e-12 AS=7.980000341079046-12 PD=6.1999999161344e-6
+PS=11.7999998110463e-6
r48 vss 0 1.0
.ENDS kcells_kor2_1_extracted
.end•
```

Appendix E.1 UNIX Manual (MAN) Page

The following source code is the man page used in UNIX, for explaining the software program and its command line.

```
.TH power 3 local
.SH NAME

.I power
\- simulates and determines dynamic and short-circuit power dissipations for designs prior to device
fabrication.
.SH SYNOPSIS

.B power
[netlist file]
[output file]
.PP

.SH DESCRIPTION

.I power
is used to simulate and determine the power dissipation in SSI, MSI, & LSI CMOS designs.
A Spice formatted netlist, produced from Cadence Design Software, or
other design tools, is imported into this program.
The necessary information is extracted and each node of the device is examined. The
total capacitance and the various energy consumption and power dissipations are calculated for each
node, based on the input test vectors. The overall power dissipation is
then calculated.
.PP
The results are stored in a matlab formatted data file. Matlab program source
code is then
appended to the data file.
.PP
.I power
requires the following command line variables:
.TP
.B netlist file
This is the original Spice extracted netlist file. It should reside in your current
working directory.
.TP
.B output file
This is the new file you would like the final results stored in, for analysis and graphing
purposes. It must have a ".m" extension.
.SH Example:
 power  cmos_cct  results.m
.PP
This program was developed at the University of Alberta as partial requirements
for an M.Sc. Graduate Thesis. The current version of power is 1.2a.
.PP
```